ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240807

Synthesis of Reward Machines for Multi-Agent Equilibrium Design

Muhammad Najib[a] **and Giuseppe Perelli**[b]

[a]Heriot-Watt University
[b]Sapienza University of Rome

Abstract. Mechanism design is a well-established game-theoretic paradigm for designing games to achieve desired outcomes. This paper addresses a closely related but distinct concept, equilibrium design. Unlike mechanism design, the designer's authority in equilibrium design is more constrained; she can only modify the incentive structures in a given game to achieve certain outcomes without the ability to create the game from scratch. We study the problem of equilibrium design using dynamic incentive structures, known as reward machines. We use weighted concurrent game structures for the game model, with goals (for the players and the designer) defined as mean-payoff objectives. We show how reward machines can be used to represent dynamic incentives that allocate rewards in a manner that optimises the designer's goal. We also introduce the main decision problem within our framework, the payoff improvement problem. This problem essentially asks whether there exists a dynamic incentive (represented by some reward machine) that can improve the designer's payoff by more than a given threshold value. We present two variants of the problem: strong and weak. We demonstrate that both can be solved in polynomial time using a Turing machine equipped with an NP oracle. Furthermore, we also establish that these variants are either NP-hard or coNP-hard. Finally, we show how to synthesise the corresponding reward machine if it exists.

1 Introduction

Over the past decade, Nash equilibrium (NE) and other game-theoretic concepts have been extensively used to analyse concurrent and multi-agent systems (see e.g., [11, 5, 34, 12, 1]). In this research, systems are modelled as games with agents acting rationally to fulfil their preferences. While preferences are often expressed qualitatively (e.g. by temporal logic formulae), many systems require more complex models to capture quantitative aspects like resource consumption, cost, or performance [13, 4, 2, 10]. Games with *mean-payoff* objectives [35] provide such richer preference modelling.

The game-theoretical analysis of mean-payoff games (MPGs) is a significant research area, especially in verifying their correctness [31, 6, 7, 29, 15, 8]. This involves checking whether a formal property is satisfied in some or all equilibrium outcomes. A pertinent question is: "what if the property is not satisfied in any equilibrium outcomes?" *Equilibrium design* [14] addresses this question. Inspired by the mechanism design paradigm [21, 17], equilibrium design offers a way to rectify equilibrium outcomes. However, unlike mechanism design, the designer in equilibrium design cannot create the game from scratch, but can only modify the incentive structures of an existing game.

In [14], the authors proposed *subsidy schemes* to introduce equilibria in concurrent MPGs satisfying some LTL formula [28]. In that setting, a subsidy scheme is modelled by a function mapping from states and players to additional rewards. If a player visits a certain state, the corresponding reward is paid to the player. In this paper, we generalise this incentive model with *reward machines* [19]. Such machines implement a reward mechanism that considers the execution history to dynamically assign rewards. Thus, at each iteration of the game, every agent receives a utility combining the original weight and an authoritative reward based on the current game state and the internal reward machine state. As we will show later (see Example 1), this generalisation allows us to obtain a more expressive model of incentive.

We consider games where each agent has a weight function over the states, with mean-payoff aggregation as their utility function. Additionally, a global weight function measures the designer satisfaction, also as a mean-payoff value over executions. We employ reward machines to improve the designer satisfaction. Intuitively, these machines reconfigure weights after each iteration, thus reshaping the set of equilibria. The objective is to improve the global payoff over the set of equilibria by a fixed amount Δ. This can be achieved *strategically* by synthesising and implementing an appropriate reward machine. To make the setting realistic, we assume the reward spent on each agent in every iteration is subtracted from the global weight, factoring the cost into the resulting global payoff.

In MPGs, infinite memory may be required to achieve optimal values [33]. As we will demonstrate later, infinite memory may be necessary to achieve the optimal global payoff value. However, since a reward machine is typically represented by a finite-state machine (specifically, a Mealy machine in this work), there may be cases where no finite-state reward machine can improve the global payoff by a given Δ. Therefore, we consider an approximate solution. In particular, we aim to find a reward machine that can improve the global payoff by a value ε-close to Δ for a given $\varepsilon > 0$. Moreover, ε can be arbitrarily small, allowing for an arbitrary level of precision.

In general, a game may have multiple equilibria. Therefore, we study the problem under both the *optimistic* and *pessimistic* settings. Specifically, we consider the problem of improving the global mean-payoff over the *best* possible NE by adopting an optimistic view that agents will select the equilibrium *most* convenient for the designer. We call this the *weak improvement* problem. Conversely, we also consider improving the global mean-payoff over the *worst* possible NE, considering the pessimistic case when agents select the

least convenient equilibrium for the designer. We call this the *strong improvement* problem. Furthermore, we classify the complexity of these problems. We show that both can be solved in $\mathsf{P}^{\mathsf{NP}} = \Delta_2^{\mathsf{P}}$ and are at least NP-hard or coNP-hard. To our knowledge, this is the first work that employs reward machines in the context of MPGs and game-theoretic equilibria.

Related work As previously mentioned, this work is closely related to [14], but it differs in several key aspects. Firstly, our incentive model is more expressive due to the use of reward machines. Furthermore, we measure the global property using a quantitative metric (i.e. mean-payoff value), as opposed to the qualitative property in [14]. In this respect, we provide a richer modelling of global preferences. Equilibrium design has a deep connection to mechanism design, but the two are not exactly the same. Typically in mechanism design, the designer is not given a predetermined game structure, but instead is required to provide one. Moreover, in mechanism design, the designer must ensure the reward structure is *incentive compatible* with respect to some *social choice function* [24] as the designer is primarily interested in the agents' payoffs (via the social choice function). This is not the case in equilibrium design, as the designer here is only interested in the global payoff, which may be orthogonal to the agents' payoffs.

On the other hand, the concept of a reward machine originated from the field of reinforcement learning (RL). Much of the existing work is within the domain of single-agent RL [18, 30, 19]. In [23], the authors explored reward machines for multi-agent RL systems. However, in this work, the reward machine is manually generated, as opposed to being automatically synthesised. [32] tackles the problem of automatically synthesising reward machines in cooperative multi-agent RL. Specifically, the reward machines are partly synthesised from Alternating-time Temporal Logic (ATL) specifications. However, this line of research focuses on RL systems, which differ from MPGs. Moreover, none of these papers consider any game-theoretical solution concepts.

Another related line of work involves designing equilibria using *norms*. Norm-based mechanism design has been studied in [9]. In particular, they studied *weak* and *strong* implementability, which are related to the problems addressed in our work in the sense that they correspond to optimistic (*"there is some good behaviour"*) and pessimistic (*"all behaviours must be good"*) assumptions. In [16, 27, 3], automata-based norms, referred to as *dynamic norms*, are considered. All of these works fall within the domain of normative systems, which is different from the setting considered in this paper. We believe that an incentive-based equilibrium design provides a complementary approach to norm-based equilibrium design. This is because in some circumstances, a norm may not be enforceable, but only incentives are possible (e.g., congestion/road pricing in the Ultra Low Emission Zone (ULEZ) in London).

2 Preliminaries

In this section we introduce the basic notions that will be used throughout the paper. We start with the definition of mean-payoff value and multi-player mean-payoff games.

Mean-Payoff For an infinite sequence $r \in \mathbb{R}^\omega$, let $\mathsf{mp}(r)$ be the *mean-payoff* value of r, that is, $\mathsf{mp}(r) = \liminf_{n\to\infty} \mathsf{avg}_n(r)$ where, for $n \in \mathbb{N} \setminus \{0\}$, we define $\mathsf{avg}_n(r) = \frac{1}{n}\sum_{j=0}^{n-1} r_j$, with r_j the $(j+1)$th element of r.

Multi-Player Mean-Payoff Game A *multi-player mean-payoff game* is a tuple $\mathcal{G} = \langle \mathrm{N}, \mathrm{Ac}, \mathrm{St}, \mathsf{s_{in}}, (\mathsf{d}_i)_{i\in\mathrm{N}}, \mathsf{tr}, (\mathsf{w}_i)_{i\in\mathrm{N}}, \mathsf{w}_g\rangle$ where

- $\mathrm{N} = \{1, \dots, n\}$, Ac, and St are finite non-empty sets of *players*, *actions*, and *states*, respectively;
- $\mathsf{s_{in}} \in \mathrm{St}$ is the *initial state*;
- $\mathsf{d}_i : \mathrm{St} \to 2^{\mathrm{Ac}} \setminus \{\varnothing\}$ is a *protocol function* for player i returning possible action at a given state;
- $\mathsf{tr} : \mathrm{St} \times \vec{\mathrm{Ac}} \to \mathrm{St}$ is a *transition function* mapping each pair consisting of a state $s \in \mathrm{St}$ and an *action profile* $\vec{\mathsf{a}} = (\mathsf{a}_1, \dots, \mathsf{a}_n) \in \vec{\mathrm{Ac}} = \mathrm{Ac}^n$, one for each player, to a successor state—we write $\vec{\mathsf{a}}_i$ for $\vec{\mathsf{a}}_{\{i\}}$ and $\vec{\mathsf{a}}_{-i}$ for $\vec{\mathsf{a}}_{\mathrm{N}\setminus\{i\}}$. For two decisions $\vec{\mathsf{a}}$ and $\vec{\mathsf{a}}'$, we write $(\vec{\mathsf{a}}_C, \vec{\mathsf{a}}'_{-C})$ to denote the decision where the actions for players in $C \subseteq \mathrm{N}$ are taken from $\vec{\mathsf{a}}$ and the actions for players in $\mathrm{N} \setminus C$ are taken from $\vec{\mathsf{a}}'$;
- $\mathsf{w}_i : \mathrm{St} \to \mathbb{Z}$ is player i's *weight function* mapping, for every player i and every state of the game into an integer number; and
- $\mathsf{w}_g : \mathrm{St} \to \mathbb{Z}$ is a *global weight function* mapping every state of the game into an integer number.

We define the *minimum* and *maximum weights* appearing in $\mathcal{G}$ as follows.

Definition 1. *For a given game $\mathcal{G}$ and its set of states* St*, define* $\mathrm{MinW}_j^{\mathcal{G}} = \min\{\mathsf{w}_j(\mathsf{s}) \mid \mathsf{s} \in \mathrm{St}\}$ *and* $\mathrm{MaxW}_j^{\mathcal{G}} = \max\{\mathsf{w}_j(\mathsf{s}) \mid \mathsf{s} \in \mathrm{St}\}$.

A *path* is an infinite sequence $\pi = \mathsf{s}_0, \mathsf{s}_1, \mathsf{s}_2, \ldots \in \mathrm{St}^\omega$ such that for each $k \in \mathbb{N}$, there is an action profile (in k-th step) $\vec{\mathsf{a}}^k \in \prod_{i\in\mathrm{N}} \mathsf{d}_i(\mathsf{s}_k)$, such that $\mathsf{s}_{k+1} = \mathsf{tr}(\mathsf{s}_k, \vec{\mathsf{a}}^k)$. We write $\pi_{\leq k}$ to denote the prefix of π up to and including s_k. Similarly, $\pi_{\geq k}$ denotes the suffix of π starting from s_k. Let $\mathsf{Paths}_{\mathcal{G}}(s)$ be the set of all possible paths in $\mathcal{G}$ starting from s.

A *strategy* for agent i is a Mealy machine $\sigma_i = (T_i, t_i^0, \mathrm{St}, \gamma_i, \rho_i)$, where T_i is a finite and non-empty set of *internal states*, t_i^0 is the *initial state*, $\gamma_i : \mathrm{St} \times T_i \to T_i$ is a deterministic *internal transition function*, and $\rho_i : \mathrm{St} \times T_i \to \mathrm{Ac}_i$ an *action function*. We say that a strategy σ_i is *valid* with respect to $\mathcal{G}$ if and only if $\rho_i(s, t_j) \in \mathsf{d}_i(s)$. From now on, we restrict our attention to valid strategies, and, unless otherwise stated, refer to them simply as strategies. We denote $\mathrm{Str}_i(\mathcal{G})$ the set of valid strategies for player i in $\mathcal{G}$. Moreover, for a given strategy σ_i and a finite sequence $\hat{\pi} \in \mathrm{St}^*$, by $\sigma_i(\hat{\pi}) \in \mathrm{Ac}$ we denote the action prescribed by the action function ρ_i of σ_i after the sequence $\hat{\pi}$ has been fed to the internal transition function γ_i. Note that the model of strategies implies that strategies have *perfect information* and *finite memory*, although we impose no bounds on memory size.

A *strategy profile* $\vec{\sigma} = (\sigma_1, \dots, \sigma_n)$ is a vector of strategies, one for each player. As with actions, $\vec{\sigma}_i$ denotes the strategy assigned to player i in profile $\vec{\sigma}$. Moreover, by $(\vec{\sigma}_B, \vec{\sigma}'_C)$ we denote the combination of profiles where players in disjoint B and C are assigned their corresponding strategies in $\vec{\sigma}$ and $\vec{\sigma}'$, respectively. We denote $\mathrm{Str}_A(\mathcal{G})$ the set of strategy profiles for the set A of agents. We also use $\mathrm{Str}(\mathcal{G}) = \mathrm{Str}_{\mathrm{N}}(\mathcal{G})$ to denote the strategy profiles for all the agents in the game. Whenever the game is clear from the context, we also simply use Str. Once a state s and profile $\vec{\sigma}$ are fixed, the game has an *outcome*, a path in $\mathcal{G}$, denoted by $\pi(\vec{\sigma}, s)$. Because strategies are deterministic, $\pi(\vec{\sigma}, s)$ is the unique path induced by $\vec{\sigma}$, that is, the sequence $\mathsf{s}_0, \mathsf{s}_1, \mathsf{s}_2, \ldots$ such that

- $\mathsf{s}_{k+1} = \mathsf{tr}(\mathsf{s}_k, (\rho_1(\mathsf{s}_k, t_1^k), \dots, \rho_n(\mathsf{s}_k, t_n^k)))$, and
- $t_i^{k+1} = \gamma_i(s_i^k, t_i^k)$, for all $k \geq 0$.

For a subset of agents $C \subseteq \mathrm{N}$ and strategies $\vec{\sigma}_C$, we say that a path π is *compatible* with $\vec{\sigma}_C$ if, for every $k \in \mathbb{N}$, there exists an action profile $\vec{\mathsf{a}}^k$ with $\mathsf{a}_i^k = \sigma_i(\pi_{\leq k})$ for each $i \in C$, such that

$\mathsf{s}_{k+1} = \mathsf{tr}(\mathsf{s}_k, \vec{\mathsf{a}}^k)$. Intuitively, π is compatible with $\vec{\sigma}_C$ if it can be generated when the agents in C play according to their respective strategies. We denote by $\mathsf{out}_{\mathcal{G}}(\mathsf{s}, \vec{\sigma}_C)$ the set of paths starting from s and compatible with $\vec{\sigma}_C$. Observe that $\mathsf{Paths}_{\mathcal{G}}(s)$ can also be written as $\mathsf{out}_{\mathcal{G}}(\mathsf{s}, \varnothing)$.

Given a game $\mathcal{G}$ and a strategy profile $\vec{\sigma}$, a path $\pi(\vec{\sigma})$ induces, for each player i, an infinite sequence of integers $\mathsf{w}_i(\pi(\vec{\sigma})) = \mathsf{w}_i(\mathsf{s}_{\mathsf{in}})\mathsf{w}_i(\mathsf{s}_1)\cdots$. Similarly, $\pi(\vec{\sigma})$ also induces such a sequence of integers for the global weight function $\mathsf{w}_g(\cdot)$. The *payoff* of player i in game $\mathcal{G}$ is $\mathsf{pay}_i^{\mathcal{G}}(\vec{\sigma}) = \mathsf{mp}(\mathsf{w}_i(\pi(\vec{\sigma})))$, and the *global payoff* of $\mathcal{G}$ is $\mathsf{pay}_g^{\mathcal{G}}(\vec{\sigma}) = \mathsf{mp}(\mathsf{w}_g(\pi(\vec{\sigma})))$. Whenever the game is clear from the context, we simply use $\mathsf{pay}_i(\vec{\sigma})$ and $\mathsf{pay}_g(\vec{\sigma})$, respectively.

Nash Equilibrium Using payoff functions, we can define the game-theoretic concept of Nash equilibrium [25]. For a multi-player game $\mathcal{G}$, a strategy profile $\vec{\sigma}$ is a *Nash equilibrium* of $\mathcal{G}$ if, for every player i and strategy σ_i' for player i, we have

$$\mathsf{pay}_i(\vec{\sigma}) \geq \mathsf{pay}_i((\vec{\sigma}_{-i}, \sigma_i'))\ .$$

We also say that $\vec{\sigma}$ is a j-fixed Nash Equilibrium [20] if $\mathsf{pay}_i(\vec{\sigma}) \geq \mathsf{pay}_i((\vec{\sigma}_{-i}, \sigma_i'))$ for every player $i \neq j$ different from the fixed j.

Let $\mathrm{NE}(\mathcal{G})$ and $\mathrm{NE}_j(\mathcal{G})$ be the set of Nash Equilibria and j-fixed Nash Equilibria of $\mathcal{G}$. We define $\mathsf{bestNE}(\mathcal{G}) = \sup_{\vec{\sigma}\in\mathrm{NE}(\mathcal{G})}\{\mathsf{pay}_g(\vec{\sigma})\}$ as the *best global payoff* over the set of possible outcomes sustained by a Nash Equilibrium in the game. Equivalently, we define $\mathsf{worstNE}(\mathcal{G}) = \inf_{\vec{\sigma}\in\mathrm{NE}(\mathcal{G})}\{\mathsf{pay}_g(\vec{\sigma})\}$ as the *worst global payoff* over the set of possible outcomes sustained by a Nash Equilibrium in the game. In the case of $\mathrm{NE}(\mathcal{G})$ is empty, in order to make the values of $\mathsf{bestNE}(\mathcal{G})$ and $\mathsf{worstNE}(\mathcal{G})$ well defined, we assume that $\mathsf{bestNE}(\mathcal{G}) = \mathsf{worstNE}(\mathcal{G}) = \mathrm{MinW}_g^{\mathcal{G}}$.

3 Reward Machines for Equilibrium Design

In this section, we introduce a type of finite state machine, called a *reward machine* (RM). A RM takes a path π as input, and outputs a sequence of vectors $\vec{v}_0, \vec{v}_1 \cdots \in (\mathbb{N}^n)^\omega$ that corresponds to the reward granted to the players at each step of the path. Formally, a RM is defined as a Mealy machine:

Definition 2 (Reward Machine)**.** *A RM is a Mealy machine $\mathcal{M} = \langle Q^{\mathcal{M}}, q_0^{\mathcal{M}}, \delta^{\mathcal{M}}, \tau^{\mathcal{M}}\rangle$, where $Q^{\mathcal{M}}$ is a finite (non-empty) set of states, $q_0^{\mathcal{M}}$ the initial state, $\delta^{\mathcal{M}} : Q^{\mathcal{M}} \times \mathrm{St} \to Q^{\mathcal{M}}$ a deterministic transition function, and $\tau^{\mathcal{M}} : Q^{\mathcal{M}} \times \mathrm{St} \to \mathbb{N}^n$ a reward function where $\tau_i^{\mathcal{M}}(q) = \tau^{\mathcal{M}}(q)(i)$ is the reward in the form of a natural number $k \in \mathbb{N}$ imposed on player i if the play visits $(s, q) \in \mathrm{St} \times Q^{\mathcal{M}}$. Sometimes, when it is clear from the context, the elements of the RM are denoted without superscripts.*

Reward Machine implementation. For a given game $\mathcal{G} = \langle \mathrm{N}, \mathrm{Ac}, \mathrm{St}, \mathsf{s}_{\mathsf{in}}, (\mathsf{d}_i)_{i\in\mathrm{N}}, \mathsf{tr}, (\mathsf{w}_i)_{i\in\mathrm{N}}, \mathsf{w}_g\rangle$, the implementation of $\mathcal{M}$ on $\mathcal{G}$ is the game

$$\mathcal{G}\dagger\mathcal{M} = \langle \mathrm{N}, \mathrm{Ac}, \mathrm{St}\times Q, (\mathsf{s}_{\mathsf{in}}, q_0), (\mathsf{d}_i^{\mathcal{M}})_{i\in\mathrm{N}}, \mathsf{tr}^{\mathcal{M}}, (\mathsf{w}_i^{\mathcal{M}})_{i\in\mathrm{N}}, \mathsf{w}_g^{\mathcal{M}}\rangle,$$

where: (i) $\mathsf{d}_i^{\mathcal{M}}(s, q) = \mathsf{d}_i(s)$, for each agent $i \in \mathrm{N}$; (ii) $\mathsf{tr}^{\mathcal{M}}((s, q), \vec{\mathsf{a}}) = (\mathsf{tr}(s, \vec{\mathsf{a}}), \delta(s, q))$; (iii) $\mathsf{w}_i^{\mathcal{M}}(s, q) = \mathsf{w}_i(s) + \tau_i(s, q)$; (iv) $\mathsf{w}_g^{\mathcal{M}}(s, q) = \mathsf{w}_g(s) - ||\tau(s, q)||$ [1].

For a given natural number $\beta \in \mathbb{N}$, a β-RM, denoted $\mathcal{M}_\beta$, is RM such that $||\tau(s, q)|| \leq \beta$ for each $(s, q) \in \mathrm{St} \times Q$. In this paper, we consider a *budget* β being fixed and restrict our attention only to β-RMs.

[1] By $||\vec{v}|| = \sum_{i\in\mathrm{N}} |v_i|$ we denote the classic Manhattan distance.

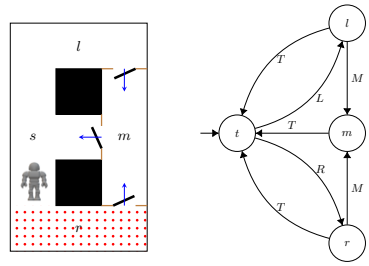

Figure 1. Graphical representation (left) and game arena (right) for Example 1.

Definition 3 (Global payoff improvement problems)**.** *For a given game $\mathcal{G}$, a budget β, and a threshold Δ. The* global payoff weak improvement *problem consists in deciding whether there exists a β-RM $\mathcal{M}$ such that:*

$$\mathsf{bestNE}(\mathcal{G}\dagger\mathcal{M}) - \mathsf{bestNE}(\mathcal{G}) > \Delta.$$

The global payoff strong improvement *problem consists in deciding whether there exists a β-RM $\mathcal{M}$ such that:*

$$\mathsf{worstNE}(\mathcal{G}\dagger\mathcal{M}) - \mathsf{worstNE}(\mathcal{G}) > \Delta.$$

Henceforth, for simplicity, we will use the term *improvement problem* to refer to the global payoff improvement problem.

At this point, it is important to note that the optimal values of bestNE and $\mathsf{worstNE}$ may not be achievable with finite-state strategies and reward machines. As such, to guarantee termination, we compute the approximate values instead. Moreover, our approach allows the values to be approximated to an arbitrary level of precision. We discuss this in detail in Section 5.

Reward Machines vs Subsidy Schemes. As previously mentioned, the reward model in this paper is a generalisation of the one considered in [14], which is referred to as a *subsidy scheme* in that paper. A subsidy scheme is defined as a function $\kappa : \mathrm{St} \to \mathbb{N}^n$. This can be trivially expressed by a reward machine $\mathcal{M} = (Q^{\mathcal{M}}, q_0^{\mathcal{M}}, \delta^{\mathcal{M}}, \tau^{\mathcal{M}})$ where $Q^{\mathcal{M}} = \{q\}, q_0^{\mathcal{M}} = q$, and for all $\mathsf{s} \in \mathrm{St}, \delta^{\mathcal{M}}(s, q) = q, \tau^{\mathcal{M}}(s, q) = \kappa(s)$. In other words, subsidy schemes belong to the subclass of "memoryless" reward machines[2]. However, there are some cases in which memory is required. To illustrate this, consider the following simple example.

Example 1. *Consider a scenario where a robot is situated in an environment shown in Figure 1 left, in which there are four locations t, l, r, m. The robot can move from one location to another and is not allowed to stay in the same location for two consecutive time steps. There are three doors separating the locations, and they can only be passed according to their respective arrows. For instance, the robot can only move from m to t through the middle door, and not the other direction. Thus, the robot can reach m from t only through l or r. However, location r is still under maintenance, and it is best to avoid passing through it. Suppose that the designer wants to incentivise the robot to deliver goods from t to m infinitely often. We can model this as a game $\mathcal{G}$ with $\mathrm{N} = \{1\}$. The game graph is shown in Figure 1 right. In each state, the actions available to the player correspond*

[2] We note that the semantics of the budget used here is slightly different to the one used in [14]. In this work budget can be thought as "capacity" of additional reward in each time step, whereas in [14] it is the total "commitment" of reward in the game.

to the outgoing edges and their respective labels. Let t be the initial state, and $\mathsf{w}_1(v) = 0$ for all $v \in \mathrm{St} = \{t, l, m, r\}$. Moreover, let $\mathsf{w}_g(t) = \mathsf{w}_g(r) = 0, \mathsf{w}_g(l) = 1, \mathsf{w}_g(m) = 2$; the designer receives reward of 2 when the robot visits m from t, furthermore, she receives extra reward of 1 when the robot uses corridor l. Suppose that $\beta = 1$. Observe that $\mathsf{worstNE}(\mathcal{G}) = 0$ corresponding to the sequence $p(t,r)^\omega$ for some finite prefix p. Suppose that we want to synthesise $\mathcal{M}$ such that given $\Delta = \frac{1}{2}$, the strong improvement problem returns a positive answer. That is, $\mathsf{worstNE}(\mathcal{G} \dagger \mathcal{M}) - \mathsf{worstNE}(\mathcal{G}) > \frac{1}{2}$.

A reward machine that satisfies the constraint in Example 1 is as follows: only give rewards of 1 when the robot visits m from l. More formally, the reward machine is shown in Figure 2 where $\tau_1^{\mathcal{M}}(q_1, m) = 1$ and $\tau_1^{\mathcal{M}}(q, t) = 0$ for all $(q, t) \neq (q_1, m)$. The set of Nash equilibria in $(\mathcal{G} \dagger \mathcal{M})$ corresponds to the sequence $p(t, l, m)^\omega$ for some finite prefix p. As such, we have $\mathsf{worstNE}(\mathcal{G} \dagger \mathcal{M}) = \frac{2}{3}$. Observe that such an incentive requires memory, since it needs to remember which path leading to m is taken by the robot. This is not possible with memoryless reward machine.

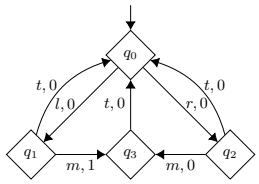

Figure 2. Reward machine $\mathcal{M}$.

4 Reward Engineering

In this and the next section, we show how to solve global payoff improvement by constructing an *auxiliary game* that allows to look at the problem as an equilibrium verification per se. More specifically, such construction regards reward machines as the strategies of a designated agent in the game, whose weight function corresponds to the global weights of the original game updated with the rewards spent on the others at each iteration. First we provide the definition of such auxiliary game, which is inspired from the constructions given in [27, 3].

Definition 4. *Given a game $\mathcal{G}$ and a budget $\beta \in \mathbb{N}$, we define its auxiliary game $\mathcal{G}' = \langle \mathrm{N}', \mathrm{Ac}', \mathrm{St}', \mathsf{s}'_{\mathsf{in}}, (\mathsf{d}_i)_{i \in \mathrm{N}'}, \mathsf{tr}', (\mathsf{w}'_i)_{i \in \mathrm{N}'} \rangle$, where (i) $\mathrm{N}' = \{0\} \cup \mathrm{N}, \mathrm{Ac}' = \mathrm{Ac} \cup \beta^n, \mathrm{St}' = \mathrm{St} \times \beta^n, \mathsf{s}'_{\mathsf{in}} = (\mathsf{s}_{\mathsf{in}}, \vec{0})$; (ii) $\mathsf{tr}'((s, \vec{v}), (\vec{\mathsf{a}}, \vec{v}')) = (\mathsf{tr}(s, \vec{\mathsf{a}}), \vec{v}')$; (iii) $\mathsf{d}'_i(s, \vec{v}) = \mathsf{d}_i(s), i \in \mathrm{N}$; (iv) $\mathsf{d}'_0(s, \vec{v}) = \{\vec{v} : ||\vec{v}|| \leq \beta\}$; (v) $\mathsf{w}'_i(s, \vec{v}) = \mathsf{w}_i(s) + \vec{v}_i$; (vi) $\mathsf{w}'_0 = \mathsf{w}_g - \|\vec{v}\|$.*

Intuitively, we are adding agent 0 to the original game $\mathcal{G}$, whose actions are n-dimensional vectors representing the possible rewards assigned to every other agent. All the other components of the auxiliary game are defined accordingly. The protocol function remains the same for every original agent, whereas the one for agent 0 prescribes that the amount of reward distributed to the agents at each iteration does not exceeds the budget β. The set of states is augmented to record the amount of reward received by each agent, which is then reflected in the corresponding weight function w'_i. Finally, the global weight function is updated by subtracting the amount of reward established by agent 0 in the current iteration.

In the next two constructions, we show how to transform a β-RM for $\mathcal{G}$ into a strategy for agent 0 and viceversa.

Construction 1 (RM to Strategy)**.** *Given a RM $\mathcal{M} = \langle Q_{\mathcal{M}}, q^0_{\mathcal{M}}, \delta_{\mathcal{M}}, \tau_{\mathcal{M}} \rangle$ of $\mathcal{G} \dagger \mathcal{M}$, we define the strategy of player 0 in $\mathcal{G}'$ as $\sigma_{\mathcal{M}} = \langle T_0, t^0_0, \mathrm{St}', \gamma_0, \rho_0 \rangle$ where $T_0 = Q_{\mathcal{M}}, t^0_0 = q^0_{\mathcal{M}}$, and the internal transition and action functions defined as*

- $\gamma_0((s, \vec{v}), t) = \delta_{\mathcal{M}}(s, t)$
- $\rho_0((s, \vec{v}), t) = \tau_{\mathcal{M}}(s, t)$

for every $(s, \vec{v}) \in \mathrm{St}'$ and $t \in T_0$.

Intuitively, the strategy $\sigma_{\mathcal{M}}$ uses the same internal states of the RM $\mathcal{M}$, while the transition and action functions of $\sigma_{\mathcal{M}}$ are defined by modifying those of $\mathcal{M}$ to match with the types required to be considered a strategy for 0 in $\mathcal{G}'$. Such construction can be reverted by carefully modifying the types, in order to move from a strategy of agent 0 in $\mathcal{G}'$ to a RM for $\mathcal{G}$, as it is shown in the following.

Construction 2 (Strategy to RM)**.** *Given a strategy $\sigma_0 = \langle T_0, t^0_0, \mathrm{St}', \gamma_0, \rho_0 \rangle$ in $\mathcal{G}'$, we define the RM for $\mathcal{G}$ as $\mathcal{M}_{\sigma_0} = \langle Q_{\mathcal{M}_{\sigma_0}}, q^0_{\mathcal{M}_{\sigma_0}}, \delta_{\mathcal{M}_{\sigma_0}}, \tau_{\mathcal{M}_{\sigma_0}} \rangle$ where $Q_{\mathcal{M}_{\sigma_0}} = T \times \beta^n, q^0_{\mathcal{M}_{\sigma_0}} = (t^0_0, \vec{0})$, and the transition and reward functions defined as*

- $\delta_{\mathcal{M}_{\sigma_0}}(s, (t, \vec{v})) = (\gamma_0((s, \vec{v}), t), \rho_0((s, \vec{v}), t))$
- $\tau_{\mathcal{M}_{\sigma_0}}(s, (t, \vec{v})) = \rho_0((s, \vec{v}), t)$

for every $s \in \mathrm{St}$ and $(t, \vec{v}) \in Q_{\mathcal{M}_{\sigma_0}}$.

We write $\pi_{\restriction \mathrm{St}}$ to denote the sequence in St^ω obtained from π by projecting the component in St and $\tau(\pi)$ the sequence in $(\mathbb{Z}^n)^\omega$ obtained from $\mathsf{w}^{\mathcal{M}}_1(\pi), \ldots, \mathsf{w}^{\mathcal{M}}_n(\pi)$.

In the following Lemma, we prove that the constructions presented above correctly translate RMs into strategies and viceversa, meaning that they make a connection between paths of $\mathcal{G} \dagger \mathcal{M}$ and outcome of $\mathcal{G}'$ when agent 0 uses the corresponding strategy and viceversa.

Lemma 1. *For a given $\mathcal{G} \dagger \mathcal{M}$ and its associated auxiliary game $\mathcal{G}'$ the following hold:*

(1) for every path $\pi \in \mathsf{Paths}_{\mathcal{G} \dagger \mathcal{M}}((\mathsf{s}_{\mathsf{in}}, q^0))$, there is a path $\pi' = (\pi_{\restriction \mathrm{St}}, \tau(\pi)) \in \mathsf{out}_{\mathcal{G}'}((\mathsf{s}_{\mathsf{in}}, \vec{0}), \sigma_{\mathcal{M}})$, and $\mathsf{w}^{\mathcal{M}}_i(\pi) = \mathsf{w}'_i(\pi')$ for all $i \in \mathrm{N}$ and $\mathsf{w}^{\mathcal{M}}_g(\pi) = \mathsf{w}'_0(\pi')$;

(2) for every path $\pi' \in \mathsf{out}_{\mathcal{G}'}((\mathsf{s}_{\mathsf{in}}, \vec{0}), \sigma_0)$, there is a path $\pi = (\pi'_{\restriction \mathrm{St}}, \delta_{\mathcal{M}_{\sigma_0}}(\pi')) \in \mathsf{Paths}_{\mathcal{G} \dagger \mathcal{M}_{\sigma_0}}((\mathsf{s}_{\mathsf{in}}, q^0))$, and $\mathsf{w}_i^{\mathcal{M}_{\sigma_0}}(\pi) = \mathsf{w}'_i(\pi')$ for all $i \in \mathrm{N}$ and $\mathsf{w}_g^{\mathcal{M}_{\sigma_0}}(\pi) = \mathsf{w}'_0(\pi')$.

Proof. We prove the first item only, as the other has a similar proof.

Observe that the path π' is uniquely identified from π. Moreover, from the definitions of $\mathcal{M}_{\sigma_0}$ and w'_0, it immediately follows that $\mathsf{w}^{\mathcal{M}}_g(\pi) = \mathsf{w}'_0(\pi')$. Therefore, we only need to prove that π' belongs to $\mathsf{out}_{\mathcal{G}'}((\mathsf{s}_{\mathsf{in}}, \vec{0}), \sigma_{\mathcal{M}})$. We do it by induction on $k \in \mathbb{N}$ by showing that every prefix $\pi'_{\leq k}$ of π' can be extended compatibly with $\sigma_{\mathcal{M}}$.

For the base case $k = 0$, we have that $\pi'_{\leq 0} = (s^0, \vec{0})$ which is trivially extendable to any path in $\mathsf{out}_{\mathcal{G}'}((\mathsf{s}_{\mathsf{in}}, \vec{0}), \sigma_{\mathcal{M}})$.

For the induction case $k > 0$, assume that $\pi'_{\leq k}$ is extendable to a path in $\mathsf{out}_{\mathcal{G}'}((\mathsf{s}_{\mathsf{in}}, \vec{0}), \sigma_{\mathcal{M}})$. Then, consider $\pi_{\leq k}$ and $\vec{\mathsf{a}}$ the joint action such that $\pi_{k+1} = \mathsf{tr}(\pi_k, \vec{\mathsf{a}})$, which exists since $\pi \in \mathsf{Paths}_{\mathcal{G} \dagger \mathcal{M}}((\mathsf{s}_{\mathsf{in}}, q^0))$. Clearly, it holds that $\pi'_{\geq k+1} = \mathsf{tr}'(\pi'_{\geq k}, (\sigma_{\mathcal{M}}(\pi'_{\geq k}, \vec{\mathsf{a}})))$, which makes $\pi'_{\leq k+1}$ extendable compatibly with $\sigma_{\mathcal{M}}$. □

Similarly to the correspondence between RMs and agent 0's strategies, a connection between strategies of any other agent i in $\mathcal{G} \dagger \mathcal{M}$ and $\mathcal{G}'$ exists. In other words, once a RM $\mathcal{M}$ and its corresponding strategy $\sigma_{\mathcal{M}}$ are fixed, every strategy σ_i for agent i in $\mathcal{G} \dagger \mathcal{M}$ can be translated into a strategy σ'_i.

Construction 3 ($\mathcal{G} \dagger \mathcal{M}$ to $\mathcal{G}'$)**.** *For a game $\mathcal{G} \dagger \mathcal{M}$ and a strategy $\sigma_i = \langle T_i, t^0_i, \gamma_i, \rho_i \rangle$ in it, we define a strategy $\hat{\sigma}_i = \langle \hat{T}_i, \hat{t}^0_i, \hat{\gamma}_i, \hat{\rho}_i \rangle$ in the corresponding game $\mathcal{G}'$ as follows:*

- $\hat{T}_i = T \times Q^{\mathcal{M}}$, *and* $\hat{t}_i^0 = (t_i^0, q_0^{\mathcal{M}})$;
- $\hat{\gamma}_i : (T \times Q^{\mathcal{M}}) \times (\mathrm{St} \times \beta^n) \to (T \times Q^{\mathcal{M}})$ *such that* $\hat{\gamma}_i((t,q),(s,\vec{v})) = (\gamma_i(t,(s,q)), \delta_{\mathcal{M}}(s,q))$;
- $\hat{\rho}_i : (T \times Q^{\mathcal{M}}) \times (\mathrm{St} \times \beta^n) \to (T \times Q^{\mathcal{M}})$ *such that* $\hat{\rho}_i((t,q),(s,\vec{v})) = \rho_i(t,(s,q))$.

By $\theta_{\mathcal{G}\dagger\mathcal{M}}(\sigma_i) = \hat{\sigma}_i$ we denote the strategy for player i in $\mathcal{G}'$ obtained from σ_i by applying the construction above.

On the other hand, once a strategy σ_0 for agent 0 in $\mathcal{G}'$ and the corresponding RM $\mathcal{M}_{\sigma_0}$ are fixed, the translation from strategies for agent i in $\mathcal{G}'$ to strategies in $\mathcal{G} \dagger \mathcal{M}_{\sigma_0}$ is possible.

Construction 4 ($\mathcal{G}'$ to $\mathcal{G} \dagger \mathcal{M}$)**.** *For a game $\mathcal{G}'$ and a strategy $\hat{\sigma}_i = \langle \hat{T}_i, \hat{t}_i^0, \hat{\gamma}_i, \hat{\rho}_i \rangle$, we define a strategy $\sigma_i = \langle T_i, t_i^0, \gamma_i, \rho_i \rangle$, in the corresponding game $\mathcal{G} \dagger \mathcal{M}$ as follows:*

- $\hat{T}_i = T \times Q^{\mathcal{M}}$, *and* $\hat{t}_i^0 = (t_i^0, q_0^{\mathcal{M}})$;
- $\hat{\gamma}_i : (T \times Q^{\mathcal{M}}) \times (\mathrm{St} \times \beta^n) \to (T \times Q^{\mathcal{M}})$ *such that* $\hat{\gamma}_i((t,q),(s,\vec{v})) = (\gamma_i(t,(s,q)), \delta_{\mathcal{M}}(s,q))$;
- $\hat{\rho}_i : (T \times Q^{\mathcal{M}}) \times (\mathrm{St} \times \beta^n) \to (T \times Q^{\mathcal{M}})$ *such that* $\hat{\rho}_i((t,q),(s,\vec{v})) = \rho_i(t,(s,q))$.

By $\theta_{\mathcal{G}'}(\hat{\sigma}_i) = \sigma_i$ we denote the strategy for player i in $\mathcal{G} \dagger \mathcal{M}$ obtained from $\hat{\sigma}_i$ by applying the construction above.

The following two lemma shows that the connection among strategies in between the games also preserves the payoff of agents.

Lemma 2. *For a given game $\mathcal{G}$, RM $\mathcal{M}$, and strategy profile $\vec{\sigma} \in \mathrm{Str}(\mathcal{G} \dagger \mathcal{M})$, it holds that*

$$\mathsf{pay}_i^{\mathcal{G}\dagger\mathcal{M}}(\vec{\sigma}) = \mathsf{pay}_i^{\mathcal{G}'}(\sigma_{\mathcal{M}}, \theta_{\mathcal{G}\dagger\mathcal{M}}(\vec{\sigma}))$$

Proof. Observe that the path $\pi = \pi(\vec{\sigma}, (\mathsf{s_{in}}, q^0))$ belongs to the set $\mathsf{Paths}_{\mathcal{G}\dagger\mathcal{M}}((\mathsf{s_{in}}, q^0))$. Moreover, by Construction 3, the path $\pi' = \pi((\sigma_{\mathcal{M}}, \theta_{\mathcal{G}\dagger\mathcal{M}}(\vec{\sigma})), (\mathsf{s_{in}}, q^0))$ is exactly the one such that $\mathsf{w}_g^{\mathcal{M}}(\pi) = \mathsf{w}_0'(\pi')$ as proved in the Item 1 of Lemma 1. This straightforwardly shows that $\mathsf{pay}_i^{\mathcal{G}\dagger\mathcal{M}}(\vec{\sigma}) = \mathsf{pay}_i^{\mathcal{G}'}(\sigma_{\mathcal{M}}, \theta_{\mathcal{G}\dagger\mathcal{M}}(\vec{\sigma}))$. □

Lemma 3. *For a given game $\mathcal{G}$, a strategy $\sigma_0 \in \mathrm{Str}_0(\mathcal{G}')$, and strategy profile $\vec{\sigma} \in \mathrm{Str}^{\mathcal{G}\dagger\mathcal{M}}$, it holds that*

$$\mathsf{pay}_i^{\mathcal{G}'}(\sigma_0, \vec{\sigma}) = \mathsf{pay}_i^{G\dagger\mathcal{M}_{\sigma_0}}(\theta_{\mathcal{G}'}(\vec{\sigma}))$$

Sketch. The proof is similar to the one of Lemma 2, with the use of Construction 4 and Item 2 of Lemma 1. □

By having the same set of payoffs, it simply follows from Lemma 1, Lemma 2, and Lemma 3, that the games $\mathcal{G} \dagger \mathcal{M}$ and $\mathcal{G}'$, where agent 0 is bound to the use of $\sigma_{\mathcal{M}}$ share the same set of Nash Equilibria.

Theorem 4. *For a given game $\mathcal{G}$ and a budget β, the two following hold:*

1. *For every β-RM $\mathcal{M}$ and strategy profile $\vec{\sigma}$ in $\mathcal{G} \dagger \mathcal{M}$, it holds that*

$$\vec{\sigma} \in \mathrm{NE}(\mathcal{G} \dagger \mathcal{M}) \text{ iff } (\sigma_{\mathcal{M}}), \hat{\vec{\sigma}}) \in \mathrm{NE}_0(\mathcal{G}').$$

2. *For every strategy profile $(\sigma_0, \vec{\sigma})$ in $\mathcal{G}'$, it holds that*

$$(\sigma_0, \vec{\sigma}) \in \mathrm{NE}_0(\mathcal{G}') \text{ iff } \theta_{\mathcal{G}\dagger\mathcal{M}_{\sigma_0}}(\vec{\sigma}) \in \mathrm{NE}(\mathcal{G} \dagger \mathcal{M}_{\sigma_0})$$

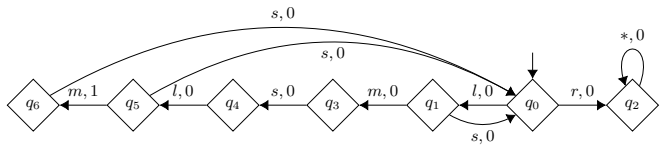

Figure 3. Reward machine $\mathcal{M}'$.

5 Solving Improvement Problems

In this section, we present a technique for solving the weak and strong improvement problems. We also demonstrate how to synthesise the RM, if it exists. With the definition of the improvement problems, it makes sense to start with the problems of computing worstNE and bestNE. To this end, we introduce *NE threshold problem* [31] that we will use as a subroutine in our algorithms. This problem asks whether there exists a NE in $\mathcal{G}$, such that the payoffs for the players fall between two vectors $\vec{x}$ and $\vec{y}$.

Definition 5 (NE Threshold Problem)**.** *Given a game $\mathcal{G}$ and vector $\vec{x}, \vec{y} \in (\mathbb{Q} \cup \{\pm\infty\})^n$, decide whether there is $\vec{\sigma} \in \mathrm{NE}(\mathcal{G})$ with $x_i \leq \mathsf{pay}_i(\vec{\sigma}) \leq y_i$ for every $i \in \mathrm{N}$.*

When the players have pure strategies, the NE threshold problem can be solved in NP [31].

We begin with the following observation. For a given game $\mathcal{G}$, it holds that $\mathrm{MinW}_g^{\mathcal{G}} \leq \mathsf{worstNE}(\mathcal{G}) \leq \mathrm{MaxW}_g^{\mathcal{G}}$ and $\mathrm{MinW}_g^{\mathcal{G}} \leq \mathsf{bestNE}(\mathcal{G}) \leq \mathrm{MaxW}_g^{\mathcal{G}}$. Moreover, it also holds that for a given $\mathcal{G}'$, we have $\mathrm{MinW}_0^{\mathcal{G}'} \leq \mathsf{worstNE}(\mathcal{G}') \leq \mathrm{MaxW}_0^{\mathcal{G}'}$ and $\mathrm{MinW}_0^{\mathcal{G}'} \leq \mathsf{bestNE}(\mathcal{G}') \leq \mathrm{MaxW}_0^{\mathcal{G}'}$. As such, by using binary search and the NE threshold problem subroutine, we can compute the values of worstNE and bestNE for $\mathcal{G}$ and $\mathcal{G}'$.

As we previously discussed, the optimal values of $\mathsf{worstNE}(\mathcal{G})$ and $\mathsf{worstNE}(\mathcal{G} \dagger \mathcal{M})$ may not be achievable with finite-state strategies and RMs. To see this, consider again Example 1. Suppose that we have a RM $\mathcal{M}'$ shown in Figure 3, where $\tau_1^{\mathcal{M}'}(q_5, m) = 1$ and $\tau_1^{\mathcal{M}'}(q, s) = 0$ for all $(q, s) \neq (q_5, m)$. Intuitively, player 1 is only given a reward of 1 after it finishes two cycles of deliveries. Clearly the set of NE still corresponds to the same sequence $p(s, l, m)^\omega$. However, since now the designer only needs to pay 1 unit for every two cycles, we have $\mathsf{worstNE}(\mathcal{G} \dagger \mathcal{M}') = \frac{5}{6}$, which is strictly greater than $\mathsf{worstNE}(\mathcal{G} \dagger \mathcal{M}) = \frac{2}{3}$ obtained by the RM in Figure 2. In fact, we can increase the number of cycles needed to be done before giving 1 unit of reward by adding more states in the RM, thus obtaining strictly greater worstNE value. Since the size of RM is not bounded, we can do this indefinitely. A similar argument can also be given for the optimal value of $\mathsf{worstNE}(\mathcal{G})$, the complete explanation can be found in the extended version [22] of this paper. Observe that by multiplying pay_g with -1, we can also use the example above to analogously reason about bestNE.

The above arguments shows that the binary search for computing the values of worstNE and bestNE may not terminate. To ensure termination, we compute approximate values instead.

Definition 6. *Given $\varepsilon > 0$, an approximate value of* worstNE *(resp.* bestNE*) is a value a such that $a - \varepsilon < o$, where o is the optimal value of* worstNE *(resp.* bestNE*). We refer to such an approximate value as ε-*worstNE *(resp. ε-*bestNE*).*

We provide Algorithm 1 for computing ε-worstNE given $\mathcal{G}$ and ε encoded in binary. The check in Line 4 corresponds to the NE threshold problem from Definition 5. Notice that the threshold vectors $\vec{x}, \vec{y}$ are not explicitly given, as we are not interested in these values. Thus,

Algorithm 1 Computing ε-worstNE

```
    input: G, ε
1:  a_1 ← MinW_g^G; a_2 ← MaxW_g^G
2:  while a_2 − a_1 ≥ ε do
3:      a' ← (a_1 + a_2)/2;
4:      if ∃σ⃗ ∈ NE(G), a_1 ≤ pay_g(σ⃗) ≤ a' then
5:          a_2 ← a'
6:      else
7:          a_1 ← a'
8:      end if
9:  end while
10: return a_2
```

we fix $x_i = \mathrm{MinW}_i^{\mathcal{G}}, y_i = \mathrm{MaxW}_i^{\mathcal{G}}$ for each $i \in \mathrm{N}$, i.e., they can be of any possible values. On the other hand, we are interested in pay_g, which in fact does not correspond to the payoff of any player. However, we can easily modify the underlying procedure for solving the problem in [31] to handle this. Specifically, by [31, Lemmas 14 and 15], we can specify an additional linear equation corresponding to the value of pay_g being in between a_1 and a', thus yielding a procedure that is also in NP. Algorithm 1 can also be used to compute ε-worstNE$(\mathcal{G}')$ with the following adaptation: Line 4 is slightly modified into $\exists \vec{\sigma} \in \mathrm{NE}_0(\mathcal{G}), a_1 \leq \mathsf{pay}_0(\vec{\sigma}) \leq a'$, that is, the NE set corresponds to the 0-fixed NE. Just as with pay_g, pay_0 is not the payoff of any player in N. Therefore, we modify the underlying procedure for the NE threshold problem using the same approach as the above.

To compute ε-bestNE, we can employ a similar technique. We make the following modification to Algorithm 1: in each iteration, instead of checking the left-half part, we check the right-half part (i.e., instead of minimising, we are *maximising*). This is done in Lines 4-8 of the algorithm by checking whether $\exists \vec{\sigma} \in \mathrm{NE}(\mathcal{G}), a' \leq \mathsf{pay}_g(\vec{\sigma}) \leq a_2$. If the check returns true, we set $a_1 \leftarrow a'$, otherwise $a_2 \leftarrow a'$. Again, as with worstNE, we slightly modify Line 4 in order to compute bestNE$(\mathcal{G}')$.

Theorem 5. *Given a game $\mathcal{G}$ (resp. $\mathcal{G}'$) and $\varepsilon > 0$, the problems of computing ε-bestNE$(\mathcal{G})$ and ε-worstNE$(\mathcal{G})$ (resp. ε-bestNE$(\mathcal{G}')$ and ε-worstNE$(\mathcal{G}')$) are $\mathsf{FP}^{\mathsf{NP}}$-complete.*

Proof. The upper bounds follows from Algorithm 1. The while loop runs in polynomial number of steps (i.e., logarithmic in $|\mathcal{G}| \cdot \frac{1}{\varepsilon}$), and in each step calls a NP oracle. Observe that ε can be arbitrarily small (i.e., arbitrary precision). For the lower bound we reduce from TSP Cost which is $\mathsf{FP}^{\mathsf{NP}}$-hard [26]. Given a TSP Cost instance $(G, c), G = (V, E)$ is a graph, $c : E \to \mathbb{Z}$ is a cost function, we construct a game $\mathcal{G}$ such that the ε-worstNE$(\mathcal{G})$ corresponds to the value of optimum tour[3]. Let $\mathcal{G}$ be such a game where

- $\mathrm{N} = V$,
- $\mathrm{St} = \{(e, v) : e \in E \wedge v = \mathsf{trg}(e)\} \cup \{(\star, \mathtt{sink})\}$,
- s^0 can be chosen arbitrarily from $\mathrm{St} \setminus \{(\star, \mathtt{sink})\}$,
- for each state $(e, v) \in \mathrm{St}$ and each player $i \in \mathrm{N}$
  - $\mathsf{d}_i((e, v)) = \{\mathsf{out}(v)\} \cup \{\star\}$ if $i = v$
  - $\mathsf{d}_i((e, v)) = \{\circ, \star\}$, otherwise;
- for each state $(e, v) \in \mathrm{St}$ and action profile $\vec{\mathrm{Ac}}$
  - $\mathsf{tr}((e, v), \vec{\mathrm{Ac}}) = (\mathsf{a}_v, \mathsf{trg}(\mathsf{a}_v))$ if $v \neq \mathtt{sink}$ and $\forall i \in \mathrm{N}, \mathsf{a}_i \neq \star$;
  - $\mathsf{tr}((e, v), \vec{\mathrm{Ac}}) = (\star, \mathtt{sink})$, otherwise;
- for each state $(e, v) \in \mathrm{St}$ and player $i \in \mathrm{N}$
  - $\mathsf{w}_i((e, v)) = |V|$, if $v = i$ and $v \neq \mathtt{sink}$,
  - $\mathsf{w}_i((e, v)) = 0$, if $v \neq i$ and $v \neq \mathtt{sink}$,
  - $\mathsf{w}_i((e, v)) = 1$, if $v = \mathtt{sink}$;
- for each state $(e, v) \in \mathrm{St}$
  - $\mathsf{w}_g((e, v)) = \max\{c(e') : e' \in E\} \cdot |V|$, if $v = \mathtt{sink}$
  - $\mathsf{w}_g((e, v)) = c(e) \cdot |V|$, otherwise;

where $\circ, \star, \mathtt{sink}$ are fresh symbols. We also set $\varepsilon = 1$. The construction is complete and polynomial to the size of (G, c).

We argue that $\lfloor \varepsilon\text{-worstNE}(\mathcal{G}) \rfloor$ is exactly the value of optimal valid tour. First, observe that for any $\vec{\sigma} \in \mathrm{NE}(\mathcal{G})$, it holds that either (1) $\pi(\vec{\sigma})$ visits every $v \in V$ (i.e., visits every city), thus a valid tour, or (2) $\pi(\vec{\sigma})$ enters $(\star, \mathtt{sink})$ and stays there forever. Case (1) holds because if $\pi(\vec{\sigma})$ does not visit $v \in V$, then $\mathsf{pay}_v(\vec{\sigma}) = 0$ thus player v will deviate to $(\star, \mathtt{sink})$ and obtain better payoff. In fact, $\vec{\sigma}$ visits each city exactly once, because otherwise, there is a player who gets payoff strictly less than 1, and deviates to $(\star, \mathtt{sink})$. Case (2) is trivially true; however, assuming that the costs are not uniform (otherwise TSP Cost becomes trivial), it cannot be a solution to ε-worstNE. Let o be the optimal tour cost, and suppose for a contradiction that $\lfloor \varepsilon\text{-worstNE}(\mathcal{G}) \rfloor < o$. Let $\hat{\vec{\sigma}}$ be a corresponding strategy profile. By the construction of $\mathcal{G}$, this means that $\hat{\vec{\sigma}}$ does not visit some cities or visits some cities more than once. However, by (1) above, $\hat{\vec{\sigma}}$ cannot be in $\mathrm{NE}(\mathcal{G})$—a contradiction. We can argue in a similar manner for $\lfloor \varepsilon\text{-worstNE}(\mathcal{G}) \rfloor > o$; it is not possible because either the corresponding strategy does not form a valid tour (and by (1) above, it is not a NE), or it is not the optimal solution to ε-worstNE; again a contradiction. Finally, since ε-worstNE approaches worstNE from the right, we have $\lfloor \varepsilon\text{-worstNE}(\mathcal{G}) \rfloor = o$.

For bestNE, we can use the same construction but with the following modification to w_g:

- $\mathsf{w}_g((e, v)) = -(\max\{c(e') : e' \in E\} \cdot |V|)$, if $v = \mathtt{sink}$
- $\mathsf{w}_g((e, v)) = -(c(e) \cdot |V|)$, otherwise;

and use similar argument as the above. □

Approximate improvement problems We define the approximate improvement problems as follows.

Definition 7 (ε-improvement problem)**.** *Given a game $\mathcal{G}$, a budget β, a threshold Δ, and ε. The Γ ε-improvement problem, with $\Gamma \in \{strong, weak\}$, decides whether there exists a β-RM $\mathcal{M}$ such that:*

$$\varepsilon\text{-}\Gamma\mathsf{NE}(\mathcal{G} \dagger \mathcal{M}) - \varepsilon\text{-}\Gamma\mathsf{NE}(\mathcal{G}) > \Delta.$$

Having the procedures for computing ε-worstNE and ε-bestNE for both $\mathcal{G}$ and $\mathcal{G}'$, we can then directly solve the ε-improvement problem with the following procedure.

1. Build the auxiliary game $\mathcal{G}'$;
2. Compute ε-ΓNE$(\mathcal{G})$ and ε-ΓNE$(\mathcal{G}')$;
3. If ε-ΓNE$(\mathcal{G}') - \varepsilon$-$\Gamma$NE$(\mathcal{G}) > \Delta$, then return "yes"; otherwise return "no".

Theorem 6. *Strong and weak ε-improvement problems are Δ_2^{P}.*

Proof. The upper bounds follow from the procedure described above. Steps 1 and 3 can be done in polynomial time, Step 2 only needs two calls to an $\mathsf{FP}^{\mathsf{NP}}$ oracle. Thus we have a decision procedure that runs in $\mathsf{P}^{\mathsf{NP}} = \Delta_2^{\mathsf{P}}$. □

[3] For auxiliary game $\mathcal{G}'$, we can easily adapt the reduction by substituting w_g with w_0.

Theorem 7. *Strong and weak ε-improvement problems are* NP*-hard and* coNP*-hard, respectively.*

Proof. To show that strong ε-improvement problem is NP-hard, we reduce from HAMILTONIAN PATH problem: given a directed graph $G = (V, E)$, is there a path that visits each vertex exactly once; this problem is NP-hard [26]. We build a game G and fix β, Δ and ε such that the strong ε-improvement problem returns yes if and only if HAMILTONIAN PATH returns yes. Given a HAMILTONIAN PATH instance $G = (V, E)$, we construct a game $\mathcal{G}$ as follows.

- $\mathrm{N} = V \cup \{n+1, n+2\}$, where $V = \{1, ..., n\}$,
- $\mathrm{St} = \{(e, v) : e \in E \wedge v = \mathsf{trg}(e)\} \cup \{(\star, \mathtt{sink}), (\star, \blacksquare), (\star, \triangle)\}$,
- s_{in} can be chosen arbitrarily from $\mathrm{St} \setminus \{(\star, \mathtt{sink}), (\star, \blacksquare), (\star, \triangle)\}$,
- for each state $(e, v) \in \mathrm{St}$ and each player $i \in \mathrm{N}$
  - $\mathsf{d}_i((e, v)) = \{\mathsf{out}(v)\} \cup \{\star\}$ if $i = v$
  - $\mathsf{d}_i((e, v)) = \{\circ, \star\}$, otherwise;
- for each state $(e, v) \in \mathrm{St}$ and action profile $\vec{\mathrm{Ac}}$
  - $\mathsf{tr}((e, v), \vec{\mathrm{Ac}}) = (\mathsf{a}_v, \mathsf{trg}(\mathsf{a}_v))$ if $v \neq \mathtt{sink}$ and $\forall i \in V, \mathsf{a}_i \neq \star$;
  - $\mathsf{tr}((e, v), \vec{\mathrm{Ac}}) = (\star, \mathtt{sink})$, if $v \neq \mathtt{sink}$ and $\exists i \in V, \mathsf{a}_i = \star$;
  - $\mathsf{tr}((e, v), \vec{\mathrm{Ac}}) = (\star, \blacksquare)$, if $v = \mathtt{sink}$ and $\mathsf{a}_{n+1} = \mathsf{a}_{n+2}$;
  - $\mathsf{tr}((e, v), \vec{\mathrm{Ac}}) = (\star, \triangle)$, if $v = \mathtt{sink}$ and $\mathsf{a}_{n+1} \neq \mathsf{a}_{n+2}$;
  - $\mathsf{tr}((e, v), \vec{\mathrm{Ac}}) = (\star, v)$, if $v \in \{\blacksquare, \triangle\}$;
- for each state $(e, v) \in \mathrm{St}$ and player $i \in \{1, ..., n\}$
  - $\mathsf{w}_i((e, v)) = |V|$, if $v = i$ and $v \notin \{\mathtt{sink}, \blacksquare, \triangle\}$,
  - $\mathsf{w}_i((e, v)) = 0$, if $v \neq i$ and $v \notin \{\mathtt{sink}, \blacksquare, \triangle\}$,
  - $\mathsf{w}_i((e, v)) = 1$, if $v \in \{\mathtt{sink}, \blacksquare, \triangle\}$;
- for each state $(e, v) \in \mathrm{St}$ and player $i \in \{n+1, n+2\}$
  - $\mathsf{w}_i((e, v)) = 0$;
- for each state $(e, v) \in \mathrm{St}$
  - $\mathsf{w}_g((e, v)) = 0$, if $v \in \{\mathtt{sink}, \blacksquare, \triangle\}$
  - $\mathsf{w}_g((e, v)) = |V|$, otherwise;

where $\circ, \star, \mathtt{sink}, \blacksquare, \triangle$ are fresh symbols. We also set $\beta = 1, \varepsilon = 1, \Delta = \frac{1}{2}$. The construction is complete and polynomial to the size of G.

Observe that $\mathsf{worstNE}(\mathcal{G}) = 0$, where the play goes to either $(*, \blacksquare)$ or $(*, \triangle)$ and stays there forever. However, with $\beta = 1$, the designer can pay player $n+1$ (resp. player $n+2$) with a payment of 1 when the play reaches $(*, \blacksquare)$ (resp. $(*, \triangle)$). Essentially, forcing $n+1$ and $n+2$ to play a matching pennies game, a game with no Nash equilibrium. Thus, the play that goes to either $(*, \blacksquare)$ or $(*, \triangle)$ no longer part of $\mathrm{NE}(\mathcal{G})$. Now, consider a run that visits each $v \in V$ exactly once, this is a Nash equilibrium. The reasoning is the same as the one provided in the proof of Theorem 5. And by construction, such a run can only be possible if and only if there is a Hamiltonian path in the corresponding graph G. Let π be such a run, now we have $\mathsf{pay}_g(\pi) = 1$ and thus, $\varepsilon\text{-}\mathsf{worstNE}(\mathcal{G} \dagger \mathcal{M}) - \varepsilon\text{-}\mathsf{worstNE}(\mathcal{G}) = 1 - 0 > \frac{1}{2}$.

The proof for weak ε-improvement problem is similar: through a reduction from the complement of HAMILTONIAN PATH. The proof is included in the extended version of this paper [22]. □

Synthesis of Reward Machines Given a game $\mathcal{G}$, a budget β, a threshold Δ, and ε, if the strong (resp. weak) ε-improvement problem returns a positive answer, then we can synthesise the corresponding RM $\mathcal{M}$ as follows. From the auxiliary game $\mathcal{G}'$, find a strategy profile $(\sigma_0, \vec{\sigma}) \in \mathrm{NE}_0(\mathcal{G}')$ such that $\mathsf{pay}_0^{\mathcal{G}'}(\sigma_0, \vec{\sigma}) = \mathsf{worstNE}(\mathcal{G}')$ (resp. $\mathsf{pay}_0^{\mathcal{G}'}(\sigma_0, \vec{\sigma}) = \mathsf{worstNE}(\mathcal{G}')$). Using Construction 2, we obtain the RM $\mathcal{M}_{\sigma_0}$ from σ_0, which corresponds to the required RM.

6 Conclusion

In this paper, we examined games where each agent had a weight function over states, with their utility determined by the mean-payoff aggregation. A global weight function was used to gauge designer satisfaction, also measured through mean-payoff value. We utilised reward machines to enhance designer satisfaction, reconfiguring weights after each iteration to reshape the equilibrium set. Our aim was to boost the global payoff over equilibria by at least a given value Δ, achieved by strategically synthesising a suitable reward machine.

Among the other results, we first demonstrated that reward machines are strictly more effective than subsidy schemes. However, we also found that in some cases, although no reward machine could improve the global payoff by the required value Δ, an ε-approximation could be found. Thus, we introduced and addressed the ε-improvement problem as a more general approach to equilibrium design.

Since multiple equilibria are possible in these games, we analysed the synthesis problem from both optimistic and pessimistic perspectives. We aimed to enhance the global mean-payoff over the best and worst possible Nash Equilibria, considering scenarios where agents select the most or least convenient equilibrium from the designer's viewpoint, respectively. We also provided complexity classifications for these problems, demonstrating that each could be solved in Δ_2^{P} and were at least NP-hard and coNP-hard.

Future work Several directions are possible from this. First, extensions of designers and agents' objectives should be considered. For example, in [14, 13] the agents' goals are represented as a combination of LTL and mean-payoff objectives, arranged in a lexicographic fashion. Also multi-valued logic such as LTL[F] are considered for rational verification [4]. It would be interesting to find out how to employ reward machines to boost the satisfaction value for this case. Last but not least, an excursion into normative systems should be considered. Although dynamic norms as defined in [16] are of the same type of reward machines [19], their implementation to games provide very different effects. On the one hand, norms disable agents' actions. On the other hand, reward machines do not strictly forbid agents to execute their actions in the game, but rather reward-incentivise those that are more convenient from the global standpoint. It would be interesting to combine the two approaches, finding the right balance between obligation and recommendation modalities.

Acknowledgements

Perelli was supported by the PNRR MUR project PE0000013-FAIR and the PRIN 2020 projects PINPOINT. He was also supported by Sapienza University of Rome under the "*Progetti Grandi di Ateneo*" programme, grant RG123188B3F7414A (ASGARD - Autonomous and Self-Governing Agent-Based Rule Design).

References

[1] A. Abate, J. Gutierrez, L. Hammond, P. Harrenstein, M. Kwiatkowska, M. Najib, G. Perelli, T. Steeples, and M. Wooldridge. Rational verification: game-theoretic verification of multi-agent systems. *Applied Intelligence*, 51(9):6569–6584, 2021.

[2] N. Alechina and B. Logan. State of the art in logics for verification of resource-bounded multi-agent systems. In A. Blass, P. Cégielski, N. Dershowitz, M. Droste, and B. Finkbeiner, editors, *Fields of Logic and Computation III - Essays Dedicated to Yuri Gurevich on the Occasion of His 80th Birthday*, volume 12180 of *Lecture Notes in Computer Science*, pages 9–29. Springer, 2020. doi: 10.1007/978-3-030-48006-6_2. URL https://doi.org/10.1007/978-3-030-48006-6_2.

[3] N. Alechina, G. D. Giacomo, B. Logan, and G. Perelli. Automatic synthesis of dynamic norms for multi-agent systems. In G. Kern-Isberner, G. Lakemeyer, and T. Meyer, editors, *Proceedings of the 19th International Conference on Principles of Knowledge Representation and Reasoning, KR 2022, Haifa, Israel, July 31 - August 5, 2022*, 2022. URL https://proceedings.kr.org/2022/2/.

[4] S. Almagor, O. Kupferman, and G. Perelli. Synthesis of controllable Nash equilibria in quantitative objective games. In *IJCAI*, pages 35–41, 2018.

[5] P. Bouyer, R. Brenguier, N. Markey, and M. Ummels. Pure nash equilibria in concurrent deterministic games. *Logical methods in computer science*, 11, 2015.

[6] R. Brenguier. Robust equilibria in mean-payoff games. In *International Conference on Foundations of Software Science and Computation Structures*, pages 217–233. Springer, 2016.

[7] L. Brice, J. Raskin, and M. van den Bogaard. Subgame-perfect equilibria in mean-payoff games. In S. Haddad and D. Varacca, editors, *32nd International Conference on Concurrency Theory, CONCUR 2021, August 24-27, 2021, Virtual Conference*, volume 203 of *LIPIcs*, pages 8:1–8:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021. doi: 10.4230/LIPIcs.CONCUR.2021.8. URL https://doi.org/10.4230/LIPIcs.CONCUR.2021.8.

[8] L. Brice, J.-F. Raskin, and M. van den Bogaard. Rational verification for nash and subgame-perfect equilibria in graph games. In *48th International Symposium on Mathematical Foundations of Computer Science (MFCS 2023)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2023.

[9] N. Bulling and M. Dastani. Norm-based mechanism design. *Artificial Intelligence*, 239:97–142, 2016.

[10] N. Bulling and V. Goranko. Combining quantitative and qualitative reasoning in concurrent multi-player games. *Auton. Agents Multi Agent Syst.*, 36(1):2, 2022. doi: 10.1007/S10458-021-09531-9. URL https://doi.org/10.1007/s10458-021-09531-9.

[11] J. Gutierrez, P. Harrenstein, and M. Wooldridge. Iterated Boolean Games. *Information and Computation*, 242:53–79, 2015.

[12] J. Gutierrez, P. Harrenstein, and M. Wooldridge. From Model Checking to Equilibrium Checking: Reactive Modules for Rational Verification. *Artificial Intelligence*, 248:123–157, 2017.

[13] J. Gutierrez, A. Murano, G. Perelli, S. Rubin, and M. Wooldridge. Nash Equilibria in Concurrent Games with Lexicographic Preferences. In *IJCAI*, pages 1067–1073, 2017. doi: 10.24963/ijcai.2017/148.

[14] J. Gutierrez, M. Najib, G. Perelli, and M. J. Wooldridge. Equilibrium design for concurrent games. In W. J. Fokkink and R. van Glabbeek, editors, *30th International Conference on Concurrency Theory, CONCUR 2019, August 27-30, 2019, Amsterdam, the Netherlands*, volume 140 of *LIPIcs*, pages 22:1–22:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi: 10.4230/LIPIcs.CONCUR.2019.22. URL https://doi.org/10.4230/LIPIcs.CONCUR.2019.22.

[15] J. Gutierrez, M. Najib, G. Perelli, and M. Wooldridge. On the complexity of rational verification. *Annals of Mathematics and Artificial Intelligence*, 91(4):409–430, 2023.

[16] X. Huang, J. Ruan, Q. Chen, and K. Su. Normative multiagent systems: A dynamic generalization. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 1123–1129. AAAI Press, 2016. ISBN 9781577357704.

[17] L. Hurwicz and S. Reiter. *Designing Economic Mechanisms*. Cambridge University Press, 2006.

[18] R. T. Icarte, T. Klassen, R. Valenzano, and S. McIlraith. Using reward machines for high-level task specification and decomposition in reinforcement learning. In *International Conference on Machine Learning*, pages 2107–2116. PMLR, 2018.

[19] R. T. Icarte, T. Q. Klassen, R. Valenzano, and S. A. McIlraith. Reward machines: Exploiting reward function structure in reinforcement learning. *Journal of Artificial Intelligence Research*, 73:173–208, 2022.

[20] O. Kupferman, G. Perelli, and M. Y. Vardi. Synthesis with Rational Environments. In *EUMAS'14*, volume 8953 of *Lecture Notes in Computer Science*, pages 219–235. Springer, 2014. doi: 10.1007/978-3-319-17130-2_15. URL https://doi.org/10.1007/978-3-319-17130-2_15.

[21] R. B. Myerson. *Mechanism design*. Springer, 1989.

[22] M. Najib and G. Perelli. Synthesis of reward machines for multi-agent equilibrium design (full version), 2024. URL https://arxiv.org/abs/2408.10074.

[23] C. Neary, Z. Xu, B. Wu, and U. Topcu. Reward machines for cooperative multi-agent reinforcement learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, page 934–942, Richland, SC, 2021. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450383073.

[24] N. Nisan et al. Introduction to mechanism design (for computer scientists). *Algorithmic game theory*, 9:209–242, 2007.

[25] M. Osborne and A. Rubinstein. *A Course in Game Theory.* MIT Press, 1994.

[26] C. Papadimitriou. *Computational complexity*. Addison-Wesley, Reading, Massachusetts, 1994. ISBN 0201530821.

[27] G. Perelli. Enforcing equilibria in multi-agent systems. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, page 188–196, Richland, SC, 2019. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450363099.

[28] A. Pnueli. The Temporal Logic of Programs. In *FOCS*, pages 46–57. IEEE, 1977.

[29] T. Steeples, J. Gutierrez, and M. Wooldridge. Mean-payoff games with ω-regular specifications. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1272–1280, 2021.

[30] R. Toro Icarte, E. Waldie, T. Klassen, R. Valenzano, M. Castro, and S. McIlraith. Learning reward machines for partially observable reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

[31] M. Ummels and D. Wojtczak. The Complexity of Nash Equilibria in Limit-Average Games. In *CONCUR*, pages 482–496, 2011. doi: 10.1007/978-3-642-23217-6_32.

[32] G. Varricchione, N. Alechina, M. Dastani, and B. Logan. Synthesising reward machines for cooperative multi-agent reinforcement learning. In *European Conference on Multi-Agent Systems*, pages 328–344. Springer, 2023.

[33] Y. Velner, K. Chatterjee, L. Doyen, T. Henzinger, A. Rabinovich, and J. Raskin. The Complexity of Multi-Mean-Payoff and Multi-Energy Games. *Information and Computation*, 241:177–196, 2015.

[34] M. Wooldridge, J. Gutierrez, P. Harrenstein, E. Marchioni, G. Perelli, and A. Toumi. Rational Verification: From Model Checking to Equilibrium Checking. In *AAAI*, pages 4184–4191. AAAI Press, 2016.

[35] U. Zwick and M. Paterson. The Complexity of Mean Payoff Games on Graphs. *Theoretical Computer Science*, 158(1):343 – 359, 1996. ISSN 0304-3975. doi: https://doi.org/10.1016/0304-3975(95)00188-3.