PONTIFICAL CATHOLIC UNIVERSITY OF RIO GRANDE DO SUL
FACULTY OF INFORMATICS
COMPUTER SCIENCE GRADUATE PROGRAM

BOUNDED MONITOR-PLACEMENT IN NORMATIVE ENVIRONMENTS

GUILHERME KRZISCH

Thesis submitted to the Pontifical Catholic University of Rio Grande do Sul in partial fullfillment of the requirements for the degree of Master in Computer Science.

Advisor: Prof. Felipe Meneguzzi

Porto Alegre
2018

Ficha Catalográfica

K94b Krzisch, Guilherme

Bounded Monitor-Placement in Normative Environments / Guilherme Krzisch . – 2018.
42 f.
Dissertação (Mestrado) – Programa de Pós-Graduação em Ciência da Computação, PUCRS.

Orientador: Prof. Dr. Felipe Meneguzzi.

1. Monitoring agent societies. 2. Normative systems. 3. Organizations. I. Meneguzzi, Felipe. II. Título.

Elaborada pelo Sistema de Geração Automática de Ficha Catalográfica da PUCRS
com os dados fornecidos pelo(a) autor(a).
Bibliotecário responsável: Marcelo Votto Texeira CRB-10/1974

Guilherme Krzisch

Bounded Monitor-Placement in Normative Environments

This Thesis has been submitted in partial fulfillment of the requirements for the degree of Master of Computer Science, of the Graduate Program in Computer Science, School of Technology of the Pontifícia Universidade Católica do Rio Grande do Sul.

Sanctioned on March 16, 2018.

COMMITTEE MEMBERS:

Prof. Dr. Duncan Dubugras Ruiz (PPGCC/PUCRS)

Prof. Dr. Ingrid Nunes (UFRGS)

Prof. Dr. Felipe Meneguzzi (PPGCC/PUCRS - Advisor)

POSICIONAMENTO DE MONITORES EM UM SISTEMA NORMATIVO

RESUMO

Normas podem ser usadas em sistemas multi-agentes para controlar o comportamento de agentes autônomos. Uma entidade autoritativa pode aplicar sanções em agentes que não estão seguindo as normas, com o objetivo de garantir que a sociedade se comporte de uma maneira desejada; isso requer a detecção de violações de normas com um mecanismo de monitoramento. A maioria das abordagens existentes para garantir o cumprimento de normas assume que o sistema pode ser totalmente observável; isso geralmente não é possível em ambientes reais. Nossa principal contribuição para endereçar esse problema é a formalização do problema de alocação de monitores em um sistema normativo multi-agente sob restrições orçamentárias. Mais especificamente, nós consideramos um sistema contendo (1) um conjunto de monitores possíveis que podem determinar o estado de porções de um domínio; (2) custos para a alocação desses monitores; e (3) um conjunto de normas que, se violadas, resultam em uma sanção. Nós procuramos identificar a combinação de monitores que maximiza a utilidade do sistema, comparando soluções aproximadas para o problema que usam diferentes heurísticas, e empiricamente demonstrando sua eficiência.

Palavras-Chave: Monitoramento em sociedades de agentes, Sistemas Normativos, Organizações e Instituições.

BOUNDED MONITOR-PLACEMENT IN NORMATIVE ENVIRONMENTS

ABSTRACT

Norms can be used in multi-agent systems to regulate behavior of self-interested agents. An authoritative entity can apply sanctions to non-compliant agents in order to ensure society functions in some desirable way, which requires the detection of norm violations with some monitoring mechanism. The majority of existing approaches to norm enforcement assumes that the system is fully observable; this is often not possible in realistic environments. Our main contribution to address this issue is the formalization of the problem of monitor placement within a normative multi-agent system under budgetary constraints. More specifically we consider a system containing (1) a set of possible monitors able to determine the state of portions of the domain; (2) costs for deploying the monitors; and (3) a set of norms which, if violated, result in a sanction. We seek to identify which combination of monitors maximizes the system's utility, evaluating approximate solutions using several heuristics, empirically demonstrating their efficiency.

Keywords: Monitoring agent societies, Normative systems, Organizations and institutions.

LIST OF FIGURES

Figure 3.1 – State-space graph of the simple problem 26
Figure 5.1 – Time Efficiency of Brute-Force and Approximate Solution, with time-out of 60 seconds 33
Figure 5.2 – Time Efficiency of both greedy solutions, smoothed using a sample of 100 data points and interpolated using splines 34
Figure 5.3 – Accuracy of approximate approaches, compared to the maximum U possible; error bars represent one standard deviation of uncertainty 35
Figure 5.4 – Accuracy of approximate approaches, compared to a brute-force solution; error bars represent one standard deviation of uncertainty 36

CONTENTS

1 INTRODUCTION **15**

2 BACKGROUND **17**

2.1 LOGIC LANGUAGE 17

2.2 PLANNING 18

2.3 NORMS 19

2.4 NORM MONITORING AND RELATED WORK 20

3 BOUNDED MONITOR-PLACEMENT PROBLEM **23**

3.1 NORM 23

3.2 MONITOR AND MONITOR PLACEMENT 24

3.3 PROBLEM 25

3.4 SCENARIO 25

3.4.1 DRINK-DRIVING DOMAIN 25

3.4.2 PLANNING PROBLEM 26

3.4.3 BOUNDED-MONITOR PLACEMENT PROBLEM 27

4 SOLUTION **29**

4.1 BRUTE-FORCE 29

4.2 MAPPING FROM NORMS TO MONITORS 30

4.3 NAIVE APPROXIMATE SOLUTION 30

4.4 GREEDY SOLUTION 31

4.4.1 NORM INDEPENDENCE HEURISTIC 31

4.4.2 ADD AND UPDATE HEURISTIC 32

5 EXPERIMENTS AND RESULTS **33**

6 CONCLUSION AND FUTURE WORK **37**

REFERENCES **39**

1. INTRODUCTION

Multi-agent systems are a powerful scalable technique to develop complex computer systems; they can be composed of both software and human agents. In an open multi-agent system, where agents can be designed by different entities and can enter and leave the system at any time, they cannot be assumed to share goals, e.g. in an e-commerce system where each user can have the goal of buying the same limited product [VdHW08]. In order for the system to obtain desirable outcomes in relation to some goals (e.g. preventing resource deadlock, avoiding destructive actions of an agent into the work of another agent), it requires a coordination mechanism, and behavior regulating *norms* [Dig99] are often used to perform this coordination.

Norms define expected behaviors of agents in a society and they often have an associated punishment if a violation occurs. While regimented norms — completely preventing an agent from violating a norm, which can be undesirable in some contexts — are widely used, a substantial body of work has shown the advantages of using approaches based on enforcement [BVDTV06, GCRASV09, Ld$^+$02, SS07]. *Norm enforcement* can be performed either by some organization or by other autonomous agents in the system [SC11]. Such approaches require a mechanism that monitors for norm compliance or violation and applies sanctions when appropriate, as behavior is no longer hard constrained and agents can continue to act with autonomy.

Approaches to norm enforcement often assume that all actions performed by an agent are observable. Such an assumption is, however, clearly unrealistic. For example, if there is a norm stating that agents cannot drive unless sober, in order to be able to detect all violations the organization needs to deploy monitors in all roads of the environment. Thus, even when physically possible, this is too costly for the organization; instead, it can decide to monitor only major roads or only at rush hours.

Our main contribution to address this issue is the formalization of the bounded monitor-placement problem, which allows the modeling of a monitoring problem in an organization under budgetary constraints. This problem uses the concept of monitors and how they should be deployed within a system, assuming that such deployments have a cost, so as to monitor the most important norms. With this approach, a designer needs to define how monitors can observe properties in the world, and how this relates to the existing norms.

Previous work on norm monitoring has modeled the problem in terms of how norms should be modified so as to be monitorable [ADL14], whether related states can be observed which may indicate upcoming norm violations [ABDL15] and norm monitoring in an environment with partial action observability [CS17].

Our approach builds on ideas from the planning literature, and in Chapter 2, we introduce the necessary concepts from this domain, and formalize the notion of a norm, reviewing and discussing related work. Chapter 3 introduces monitors and formally describes the problem we are addressing, situating the problem within a motivating domain. In Chapter 4 we develop solutions to the problem, presenting results based on a set of experiments in Chapter 5. We conclude in Chapter 6, indicating directions for future work.

2. BACKGROUND

In this chapter we introduce concepts from first-order language, which is used as a base both to formalize the environment as a state-transition system using planning and to formalize norms that are applicable to this system. In this way we can leverage the existing dataset of domains and problems of the planning literature as well as have a model to abstract the real world.

2.1 Logic Language

First, we introduce the basic components of this language in the following definitions. These definitions comprise a first-order language $\mathcal{L}$ containing an infinite set of symbols for constants $(c, c_1, c_2, c_3, ...)$, variables $(x, x_1, x_2, x_3, ...)$, functions $(f, f_1, f_2, f_3, ...)$ and predicates $(p, p_1, p_2, p_3, ...)$.

DEFINITION 1 (TERM). *A term τ can be a variable; a constant; or a function $f(\tau_1, \tau_2, ..., \tau_n)$, where f is a n-ary function symbol applied to n terms.*

DEFINITION 2 (PREDICATE). *A predicate ϕ, represented as $p(\tau_1, \tau_2, ..., \tau_n)$, is composed of a symbol p from the predicate list and zero or more terms.*

A predicate is ground if it does not contain variables, and we denote as $|\mathcal{L}|$ the number of ground predicates in this language.

DEFINITION 3 (FORMULA). *A first-order formula Φ is recursively defined as $\Phi \wedge \Phi' \mid \neg\Phi \mid \phi$.*

We assume the usual equivalences:

- $\Phi \vee \Phi' \equiv \neg(\neg\Phi \wedge \neg\Phi')$
- $\Phi \rightarrow \Phi' \equiv \neg\Phi \vee \Phi'$
- $\Phi \leftrightarrow \Phi' \equiv (\Phi \rightarrow \Phi') \wedge (\Phi' \rightarrow \Phi)$

In this work we do not use quantifiers; we assume that all first-order formulas are finite. Additionally, we assume a first-order inference mechanism, able to determine if a given formula Φ can be inferred from another formula Φ', represented as $\Phi \models \Phi'$. We also assume the following notational equivalence: $\{\Phi_1, \Phi_2, ..., \Phi_n\} \equiv (\Phi_1 \wedge \Phi_2 \wedge ... \wedge \Phi_n)$ (i.e. that lists of formulas are implicitly conjunctions). In the following, well-formed atomic formulas are referred as *atoms*.

Having this formal framework, in the following sections we introduce definitions related to classical planning and related to norms.

2.2 Planning

Planning is concerned with finding courses of actions to achieve objectives, given a known initial state. We build on classical planning, which assumes finite, fully observable and deterministic systems, and adapt the definitions from Ghallab *et al.* [GNT35, Ch. 2] and from Meneguzzi *et al.* [MDS15].

DEFINITION 4 (STATE). *A state is a finite set of ground atoms in a first-order language $\mathcal{L}$. We use the closed-world assumption, i.e. if a state does not specify a predicate, then this predicate does not hold in that state.*

The execution of an action by an agent changes the state of the environment; an action is an instantiation of a planning operator. These concepts are formalized as follows.

DEFINITION 5 (OPERATOR). *An operator is a triple $o = \langle name(o), pre(o), eff(o) \rangle$, where $name(o)$ is the unique description of o. $pre(o)$ and $eff(o)$ are set of atoms, which represent the planning operator's preconditions and effects respectively.*

DEFINITION 6 (ACTION). *An action a is a ground instance of a planning operator, and is applicable in state s if $s \models pre(a)$. The result of applying action a to state s is a new state s', such that $s' = (s / eff^{-}(a)) \cup eff^{+}(a)$, where eff^{-} is the set of negated predicates and eff^{+} is the set of positive predicates.*

We specify the dynamics of our multi-agent system in terms of a transition function following classical planning semantics in Definition 7, and the initial and goal states of the system in terms of planning problem instances in Definition 8.

DEFINITION 7 (PLANNING DOMAIN). *A planning domain in $\mathcal{L}$ is a state-transition system $\Sigma = \langle S, A, \gamma \rangle$, where $S \subseteq 2^{|\mathcal{L}|}$ is a subset of all possible states; A is the set of all ground instances of planning operators; $\gamma(s, a)$ is a state-transition function defined if $a \in A$ and a is applicable to $s \in S$.*

Note that S is closed under γ, i.e., given a state $s \in S$, all states reachable from applying action a in s are also in S. Therefore, although the number of possible states in $\mathcal{L}$ is equal to $2^{|\mathcal{L}|}$, the number of reachable states is smaller.

DEFINITION 8 (PLANNING PROBLEM). *A planning problem is defined as a triple $\mathcal{P} = \langle \Sigma, s_0, g \rangle$, where $s_0 \in S$ is the initial state of the problem and g, the goal, is a set of ground predicates.*

A solution for a planning problem is a sequence of actions, formalized in Definition 9.

DEFINITION 9 (PLAN). *A plan π is a sequence of actions $\langle a_1, a_2, ..., a_n \rangle$ that modifies the initial state s_0 into successive states $\langle s_1, s_2, ..., s_n \rangle$. The plan is a solution to a planning problem if $s_n \models g$, and it is optimal, denoted as π^*, if no other plan contains fewer actions.*

2.3 Norms

Norms are sets of behaviors that are imposed to agents in a society. In closed societies, when agents belong to the same organization and this organization controls the behavior of individual agents, we have regimented norms. This type of norms does not allow agents to perform actions that violate norms, as they are designed by the same entity with a common interest. In contrast, in open and dynamic societies, self-interested agents cannot be assumed to share the same set of goals, because they do not necessarily belong to the same organization; this becomes a problem when there are conflicts between goals of individual agents, and their corresponding behaviors start to impact negatively one another. In this context, enforced norms can be used to regulate and coordinate behavior [LMM+13]. In this work we are interested in this second type of norms, where agents can violate them, and in the resulting problem of how to detect such violations in a monitoring system with limited resources.

A classical example of a behavior that impacts the agents, and thus their society, is whether agents drive on the right or on the left side of the road. If each agent follows a different behavior, the number of crashes increases. To solve this problem, countries have different norms and associated sanctions to violating agents; this have two main advantages: first, it tries to enforce common behavior, minimizing the negative impact of undesirable behaviors in the overall efficiency of the society. Secondly, individual agents can reduce the amount of information needed to be processed, as indicated by [Eps01]; agents in this case do not need to reason in which side of the road to drive, they can just follow the established norm.

Generalizing from the previous example, we state that norm violations have an undesirable impact on the society, as encoded by an associated cost. To dissuade agents from violating norms, when such a violation is detected, a penalty is applied to the violating agent. We do not consider the nature of this penalty, which is the system designer responsibility, assuming instead that it is sufficiently large to prevent the agent from repeatedly violating this norm and that the damage to society is mitigated by its application. Examples of the nature of punishment include approaches that use emotions [FvSM06, SP+01], reputation loss [CCP+98, MMH02, RSGJ03] and loss of utility [DMSC00] in order to apply sanctions.

Researchers employ different formalizations of norms [DGMT09, JS93, VKN09]. One common concept used across all works is the deontic modality of the norm; this defines if a norm is an obligation (behavior that must be followed by agents), a permission (an exhaustive list of allowed behaviors - and thus all behaviors not mentioned are prohibited, or it can be assumed that every behavior not prohibited is permitted) or a prohibition (behavior that must be avoided).

A violated norm has an undesirable impact on the society. To dissuade agents from violating norms, when such a violation is detected, an enforcer applies a penalty to the agent. We must therefore consider how to monitor these norms to detect violations, which is the subject of the next section.

2.4 Norm Monitoring and Related Work

According to Savarimuthu et al. [SC11] there are five main topics on norm research: creation, identification, spreading, enforcement and emergence. While the first two are related to norm formation, i.e. how they are created and identified by other agents in the society, the last three are concerned with norm propagation, i.e. how norms spread and are assimilated. The norm enforcement component is usually responsible to apply sanctions to violator agents, in order to maintain a norm active after it has been spread in a given society.

Work on normative systems often assumes that the normative enforcement component can detect all violations, along with the violator agent [ERRAA04, GCNRA05, GCRASV06, KN02, MFM+09, yLLd06]. This is a rather strong assumption, because it assumes it is possible to observe all actions that may violate a norm and to correctly tie this violation to an agent. This is unrealistic for three reasons. First, in large systems, monitoring cost can be very high [Sti74, SA85]. Second, the environment is almost never fully observable, i.e. there are portions of the environment that monitors cannot realistically access. Third, it is not always possible to distinguish which agent triggered the norm violation.

Some authors partially dropped these assumptions in recent work. Alechina *et al.* [ABDL15] models this as a set of queries that a monitor can ask in a state, i.e., a monitor may not able to distinguish between two different states. They then approximate an ideal norm to be optimal given the observation capabilities of a monitor. In other words, their problem modifies the set of norms to a new set of approximate norms that can be optimally monitored given a set of monitors and queries. However, they neither consider deployment and monitoring costs nor the problem of distinguishing the violator agent.

Alechina *et al.* [ADL14] define norms in Linear Temporal Logic (LTL) formulas. They introduce the concept of a guard, which uses lookahead mechanisms to detect future norm violations. One key component is the size of this lookahead window, which is bounded to reduce the amount of computation in the future (they have complete knowledge of the past). In order to increase their monitoring capabilities, they increase the window size, thus also impacting the computation cost.

Bulling *et al.* [BDK13] include the concept of monitors and combination of monitors. They specify monitors and norms using LTL-formulas, and focus on exploring its properties and relations. Their current framework does not include norm or monitor costs, but they intend to add these concepts in future work.

Criado *et al.* [CS17] takes a different approach and deals with partial observability concerning actions. In order to detect violations it first reconstructs a partial action sequence before enforcing norm compliance. In Testerink *et al.* [TDB16] the problem of enforcement is modeled and addressed using control automata with distributed controllers. Balke *et al.* [BDVP13] uses simulation to empirically find the proportion of enforcement agents that can yield the best results while enforcing a set of norms; their enforcer agents move randomly in a wireless mobile grid environment, without coordination. Finally, Mouden *et a.l* [EMWG10] gives theoretical results about the evolution of enforcement and coordination, and their interactions in this process.

The work surveyed thus far [ABDL15, ADL14, BDK13], approach the problem from the perspective of either changing the norms or exploring the relations and properties of monitors and norms. Their main limitation is that they tend to not consider limitations in cost and budget of the monitoring system. In the next chapter we propose the bounded monitor-placement problem, which intends to fill this gap.

3. BOUNDED MONITOR-PLACEMENT PROBLEM

In this chapter we formalize the bounded monitor-placement problem. Acting as the object of our problem, we provide two different forms of norm formalization, which have different expressiveness levels. Then, we introduce the concept of monitors and monitor placements, which are used as the basic components in our problem. Finally, in the last section we provide a scenario with a working example of a problem instance.

3.1 Norm

The first, and less expressive norm formalization, is a *simple conditional norm* in Definition 10; it concerns actions performed in single states.

DEFINITION 10 (SIMPLE CONDITIONAL NORM). *A norm is a tuple $n = \langle \mu, \chi, \rho, C \rangle$, where:*

- *$\mu \in \{obligation, prohibition\}$ represents the norm's modality;*
- *χ is a well-formed ground formula that represents the* context *to which a norm applies, i.e. a norm is applicable in state s if $s \models \chi$;*
- *$\rho \in A$ represents the* object *of the norm's modality;*
- *C is the cost or penalty* to the society *which occurs if the norm is violated.*

We refer to each of these items for a given norm n as μ_n, χ_n, ρ_n and C_n respectively.

EXAMPLE 1. *The following simple conditional norm requires an agent to drive on the left side of the road if they are in England. An agent violating this norm causes harm to the society worth 20 units of utility.*

$$n = \langle obligation, at(England), driveLeft(a, b), 20 \rangle$$

The second norm formalization is more expressive and deals with sequences of states. We adapt the definition from [MMO+09], represented in Definition 11.

DEFINITION 11 (CONDITIONAL NORM). *A norm is a tuple $n = \langle \mu, \alpha, \beta, \tau, C \rangle$, where:*

- *$\mu \in \{obligation, prohibition\}$ represents the norm's modality;*
- *α and τ are well-formed ground formulas that represent, respectively, the activation and expiration conditions of the norm;*

- *β is a well-formed ground formula that represents the object of the norm's modality;*
- *C is the cost or penalty* to the society *which occurs if the norm is violated.*

We refer to each of these items for a given norm n as μ_n, α_n, τ_n, β_n and C_n respectively.

A *conditional norm* stays active from the state of the world where its activation condition α holds until some later state where its expiration condition τ holds. In case of an obligation norm, β describes when this norm is being complied with, while for a prohibition norm it describes when it is being violated. Thus, the violation of an active obligation norm occurs when β does not hold, while the violation of an active prohibition norm occurs when β holds.

EXAMPLE 2. *The following conditional norm requires an agent to drive on the left side of the road if they are in England. An agent violating this norm causes harm to the society worth 20 units of utility.*

$$n = \langle obligation, at(England), drivingLeft(a, b), not\ at(England), 20\rangle$$

3.2 Monitor and Monitor Placement

A monitor represents a single component in the system that is able to determine the truth value of a boolean combination of predicates, having an associated cost. Formally, we define a monitor as follows.

DEFINITION 12 (MONITOR). *A monitor $m = \langle P, D\rangle$ consists of a well-formed ground formula P, and a* deployment cost *$D \in \mathbb{R}$. A monitor is able to determine the truth value of P at cost D. We refer to the formula of a monitor m as P_m, and to its cost as D_m.*

A set of monitors can be used to monitor more complex combinations of predicates. Some monitors can conceivably detect the status of a predicate related to multiple norms, or when combined, can be used to determine the status of a norm that individual monitors cannot.

DEFINITION 13 (MONITOR PLACEMENT). *A monitor placement is a set* **M** *of monitors able to determine the truth value of a boolean combination of predicates **P**, where $\boldsymbol{P} = \bigcup_{\forall m \in \boldsymbol{M}} P_m$ consists of a combination of predicates of each monitor. It can monitor a set of norms **N**, s.t. $n \in \boldsymbol{N}$ iff*

- *$\boldsymbol{P} \models \chi_n$ and $\boldsymbol{P} \models pre(\rho_n)$ for simple conditional norms;*

- ***P*** $\models \alpha_n$*,* ***P*** $\models \tau_n$ *and* ***P*** $\models \beta_n$ *for conditional norms.*

The cost ***C*** *of a monitor placement is* $\sum_{m \in M} D_m$*, while the utility* ***U*** *is* $\sum_{n \in N} C$*, where N is the set of norms detected by this placement.*

3.3 Problem

By introducing the concept of an available budget, we define our problem of placing monitors in a system as follows.

DEFINITION 14 (BOUNDED-MONITOR PLACEMENT PROBLEM). *A bounded-monitor placement problem is encoded as a triple* $\langle M, N, B \rangle$ *where M is a set of monitors, N is a set of norms, and* $B \in \mathbb{R}^{+}$ *is a budget.*

A solution to the problem is a monitor placement MP, consisting of a subset of the set of monitors M, such that its cost is smaller than or equal to the budget.

The size of the state-space for a bounded-monitor placement problem is exponential in relation to the size of the set of monitors M. More specifically, it has a size of $2^{|M|}$. Therefore, the search of a solution for this problem is also exponential in the worst case.

We may also identify some special cases of the problem. For example, when the set of available monitors contains all possible combinations of predicates, then all norms can be monitored (given a sufficient budget). Next, we provide a concrete example of the bounded-monitor placement problem.

3.4 Scenario

To motivate and exemplify the bounded-monitor placement problem, in what follows, we introduce an example scenario and a related problem.

3.4.1 Drink-driving Domain

The domain consists of a city environment where it is possible to move between locations by driving a car. Specific locations of the city contain bars, which agents can enter, possibly drink at, and exit for an alcohol fueled drive. An agent becomes *drunk* after drinking and *not drunk* after sleeping.

There are five predicates in this domain: *at*(l) to represent an agent's location; *inBar*(b) when an agent is inside a bar b; *in*(b, l) to indicate that a bar b is located at a specific city location *l*; *connected*(*l*1, *l*2) when there is a road between two city locations *l*1 and *l*2; and *drunk*() to indicate when an agent is drunk.

Agents can perform five distinct actions in this environment: *move*(*l*1, *l*2) from location *l*1 to location *l*2; *enter*(*l*, *b*) to represent an agent entering a bar *b* located in location *l*; *exit*(*l*, *b*) to represent an agent exiting from bar *b* to location *l*; *drink*(*b*), when an agent drinks at bar *b* and becomes drunk; *sleep*(*l*) when an agent sleeps at location *l*, and becomes not drunk. In this domain, predicates *in*(*b*, *l*) and *connected*(*l*1, *l*2) never change, i.e., no operator changes them, and thus they are *rigid predicates*; the remaining three predicates are *mutable predicates*.

Having formalized the planning domain, we proceed to describing a planning problem in this domain.

3.4.2 Planning Problem

A simple problem in this domain consists of two locations (*a*, *b*) connected to each other, represented by *connected*(*a*, *b*) and *connected*(*b*, *a*); one bar (*barA*) located in location (*a*), represented by predicate *in*(*barA*, *a*). One possible initial state for a single agent is specified as follows.

$$\begin{aligned} s_0 = \langle & connected(a, b), connected(b, a), \\ & in(barA, a), not\ drunk(), at(a), not \\ & at(b), not\ inBar(barA)\rangle \end{aligned}$$

The number of ground mutable predicates in this problem is 4.

The state-space graph of this problem is shown in Figure 3.1. Note that each node represents a reachable state; we do not represent all possible states of the problem. The label of each node contains only mutable predicates evaluated to true in that state, in order to keep graph legible.

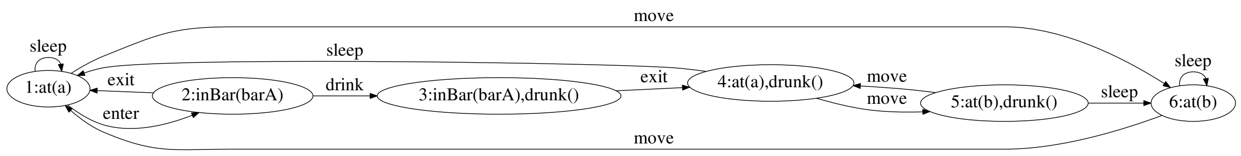

Figure 3.1 – State-space graph of the simple problem

A possible goal here could be $g = \langle at(b), drunk() \rangle$, which is only achievable by state node 5 in the graph if Figure 3.1. A possible optimal plan is then as follows:

$$\pi^* = \langle enter(a, barA), drink(barA), exit(a, barA), move(a, b) \rangle$$

There is a norm in this environment that states that it is forbidden to drive while drunk, in order to avoid collisions. We can state this norm as simple conditional norms n_1 and n_2, based on Definition 10.

$$n_1 = \langle prohibition, drunk(), move(a, b), 10 \rangle$$
$$n_2 = \langle prohibition, drunk(), move(b, a), 10 \rangle$$

Plan π^* violates the norms in this domain when, in state 4, the agent performs action *move*(a, b) while *drunk*. The only other possible violation, and which does not occur in this plan, is to perform action *move*(b, a) while *drunk* in state 5.

If instead we want to use conditional norms based on Definition 11, we can state these norms as norms n_3 and n_4. For this, a new predicate *moving* must be added, representing when an agent is currently moving from one location to another.

$$n_3 = \langle prohibition, drunk(), moving(a, b), not\ drunk(), 10 \rangle$$
$$n_4 = \langle prohibition, drunk(), moving(b, a), not\ drunk(), 10 \rangle$$

For this problem, the *simple conditional norm* is able to capture the intended prohibition, i.e. to forbid driving while drunk. However, norms with a temporal component, e.g. you cannot drive while drunk from the moment you enter in location *x* until the moment you enter in location *y*, can only be expressed using the *conditional norm*.

3.4.3 Bounded-Monitor Placement Problem

Now that we have the planning domain and the problem defined, we can state the bounded-monitor placement problem $\mathcal{B} = \langle M, N, B \rangle$, where $N = \{n_1, n_2\}$. We assume the following set of possible monitors:

$$
\begin{aligned}
m_1 =& \langle \{at(a)\}, 4\rangle \\
m_2 =& \langle \{at(b)\}, 4\rangle \\
m_3 =& \langle \{drunk()\}, 3\rangle \\
m_4 =& \langle \{at(a), drunk()\}, 6\rangle \\
m_5 =& \langle \{at(b), drunk()\}, 6\rangle
\end{aligned}
$$

If we have an unlimited budget, i.e. $B = \infty$, one of the possible solutions (and indeed an optimal solution) is the set of all possible monitors with cost 23. The problem as described above has a solution that achieves its maximum possible observability, i.e. it can observe all predicates from the set of monitors. This problem becomes harder when we limit the total budget available, so if the total budget is $B = 12$, the above solution becomes too costly, and we need to find a more restricted monitor placement. Example of restricted monitor placement include $MP_1 = (\{m_4, m_5\})$ or $MP_2 = (\{m_1, m_2, m_3\})$. Note that we can monitor norm n_1 with either monitor m_4 or with monitors m_1 and m_3. The same is true for norm n_2: it can be monitored with monitor m_5 or with monitors m_2 and m_3. However, to find the optimal solution to this is problem, it is not sufficient to simply choose the set of monitors with the least cost that is able to monitor a given norm. For example, for $D = 11$ a solution containing monitors m_4 and m_5 is too costly; instead, monitors m_1, m_2 and m_3 are able to monitors both norms, while having cost $\mathbf{C} = 11$.

4. SOLUTION

We developed five different approaches to generate monitors in partially observable environments using limited resources, which we detail in what follows. We start by describing a brute-force approach to generate all optimal solutions; first by exploring all of the solution search space in Section 4.1, and then another approach based on a mapping from norms to monitors in Section 4.2. Then we describe heuristics which find approximate solutions in linear time, consisting of a naive solution in Section 4.3, and another solution exploiting features of the problem in Section 4.4. In the next chapter we compare and evaluate these approaches with experiments.

For the proposed solutions we are assuming, without loss of generality, that each norm has a conjunction of predicates in its definition. We can make a transformation, of a general norm n into multiple norms n' that follows this assumption, with these steps:

1. Simple Conditional Norm
   (a) Convert the well-formed ground formula from the norm context χ to Disjunctive Normal Form (DNF)
   (b) For each clause c in the DNF formula, create a norm n' with context $\chi = c$
2. Conditional Norm
   (a) Convert the well-formed ground formula from the norm activation α, expiration τ and modality β to Disjunctive Normal Form (DNF)
   (b) For each clause c in the DNF formula, create a norm n' with a well-formed ground formula c

After this transformation, we have $|C|$ new norms, which can be used as input for the proposed solutions.

4.1 Brute-force

A brute-force approach is a trivial solution to this problem. It considers all possible combinations of available monitors and returns the best one, i.e. it searches the entire problem state-space. We can clearly see that this is impractical for large problems, as its complexity increases exponentially given the size of the possible monitors set input. More specifically, it has a time complexity of $O(2^{|M|})$. We use this approach as a baseline against which we compare the remaining heuristics.

4.2 Mapping from norms to monitors

The main drawback of the brute-force approach is that it includes a large number of irrelevant solutions while enumerating all possible solutions. We can minimize this problem by computing a mapping from norms to monitor placements, i.e., for each norm find the set of all monitor placements capable of monitoring it. For example, for the problem from Section 3.4.3, this mapping is as follows:

- $n_1 \rightarrow \{m_1, m_3\}$ *OR* $\{m_4\}$
- $n_2 \rightarrow \{m_2, m_3\}$ *OR* $\{m_5\}$

With this mapping, we can compute possible solutions by choosing one monitor placement for each norm. Generalizing, we have $\prod_{i=1}^{|N|}(|\mathbf{MP}_{n_i}| + 1)$ possible solutions, where $\mathbf{MP}_{n_i}$ is the set of monitor placements able to monitor norm n_i; since we also need to consider the impracticality to monitor a given norm (when there is no available budget), we need to add the empty monitor placement set. In the best case we have only one possible monitor placement for each norm, and in the worst case we have all possible combinations of available monitors for for each norm. Therefore, the number of solutions ranges from $2^{|N|}$ to $2^{|M|*|N|}$. This shows that for medium and large problems these approaches do not scale. To try to overcome the time limitation we can use approximate methods to find non-optimal solutions, which we explore in the following sections.

4.3 Naive Approximate Solution

In order to improve the run-time over the brute-force approach described in Section 4.1, we introduce a simple approximate solution whose purpose is to serve as a a baseline to compare the accuracy of the other two approximate solutions. This solution does not use any special data structure; it iterates over monitors ranked by their expected probability of detecting norm violations. In order to rank these monitors it uses the number of norms that a single monitor can partially detect, i.e., a monitor can partially detect a norm if it has at least one predicate of the norm's context or of the preconditions of the norm's action ρ. The intuition here is that choosing monitors that can locally partially observe several norms leads to a final monitor placement that can detect many existing norms.

This approach, however, does not capture essential parts of the problem. First, it does not take into consideration the norm's cost and monitor's cost. Second, it has an overly strong assumption that joining monitors that can partially detect norms (i.e. that can only detect a subset of the corresponding norm predicates, not being able to detect a norm violation) will lead to a monitor placement that can detect the problem norm violations. Thus, we use the mapping data structure introduced earlier to improve on both approaches by using greedy search in Section 4.4.

4.4 Greedy Solution

We propose two approaches with different heuristics using the mapping structure introduced in Section 4.2. By using a heuristic to rank the best monitor placements, we can avoid searching through an exponential solution search space and perform a greedy search. More specifically, we select the best monitor placement candidate of each norm. The resulting heuristic sacrifices optimality for efficiency, running in linear time.

```
1: function Find Approximate Solution(N:Norms)
2: build mapping from norms to monitor placements
3: $currentMP \leftarrow \{\}$
4: while hasBudget do
5:     $n \leftarrow extractMaxNorm(N)$
6:     $mp \leftarrow getMinMP(n)$
7:     $currentMP \leftarrow currentMP \cup mp$
8: end while
9: return $currentMP$
```

Algorithm 4.1 – Greedy algorithm

Our base algorithm structure for both heuristics is described in Algorithm 4.1. It starts by building a mapping between norms and monitor placements in Line 2, as described in Section 4.2. After this, it adds monitors to an initially empty monitor placement *currentMP* in Line 3, while there is still available budget (Line 4). At each iteration, it chooses one norm (Line 5), gets one monitor placement that is able to monitor this norm (Line 6), and adds the monitor placement to the *currentMP* (Line 7). We now describe two heuristics to speed up this Algorithm 4.1.

4.4.1 Norm Independence Heuristic

Our first heuristic considers norms to be monitored completely independently of each other when choosing monitors in order to substantially prune the search space of the problem. We first need to define which norm *extractMaxNorm(N)* chooses; for this, we select the norm with the highest penalty, in order to increase the value of **U** (the sum of each individual norm penalty):

$$\arg\max_{n \in N} \ C_n$$

The other decision is which monitor placement is selected by the *getMinMP(n)* method; for this, we select the one with the lowest cost, as it is the better monitor placement able to monitor norm *n*:

$$\arg\min_{MP \in \mathbf{MP}_n} \ \mathbf{C}_{MP}$$

This approach does not consider the intersection of monitors that are able to monitor multiple norms; it selects monitor placements independently of the others. Consequently, we improve this solution in what follows by introducing the concept of *current cost* to try to find a better approximate solution for our problem.

4.4.2 Add and Update Heuristic

The structure of this heuristic is similar to the previous approach; but instead of using the cost of each monitor placement as the metric to select the one with lowest budget, we use its *current cost*. This *current cost* of a given monitor placement *MP* is computed as Formula 4.1, meaning that we only consider the cost of monitors that were not already selected in a previous iteration (and thus not in the set of monitors of *currentMP*).

$$\mathbf{CC}_{MP} = \sum_{m \in M_{MP} \wedge m \notin M_{currentMP}} D_m \tag{4.1}$$

$$\arg\min_{MP \in \mathbf{MP}_n} \ \mathbf{CC}_{MP} \tag{4.2}$$

$$\arg\max_{n \in N} \ \frac{C_n}{\mathbf{CC}_{getMinMP(n)}} \tag{4.3}$$

Then the *getMinMP(n)* method is implemented as Formula 4.2, which we use to compute the next norm to be monitored using the *extractMaxNorm(N)* method, implemented as Formula 4.3. Basically it chooses the norm with the highest value of the ratio between its penalty and the lowest *current cost* of the set of monitor placements. Using the concept of *current cost* we are disregarding the costs of monitors that have already been chosen in past iterations, yielding better estimates. In the next section we describe our experiments and the results for the different approaches proposed in this section.

5. EXPERIMENTS AND RESULTS

To empirically evaluate our approaches, we automatically generate sets of norms and sets of monitors with increasing complexity. The experiments were performed on a computer equipped with an Intel Core i7-6700 and 16 GB of RAM running a 64-bit version of Ubuntu 16.04, with an OpenJDK 64-Bit Server Java Virtual Machine[1]. Our main tested domain is the *drink-driving* domain from Section 3.4; we also tested with *blocksworld, depots, dwr, easy_ipc_grid, gripper, logistics* and *robby* domains, from different planning competitions [Bac01, EH21, LF03].

There are two metrics to consider when analyzing results: time performance and accuracy. When comparing time performance we use the brute-force approach as a baseline for the approximate approaches. In Figure 5.1 we can see that the brute-force approach becomes intractable for relative small number of norms, while the approximate approaches remains linear as the number of norms increases.

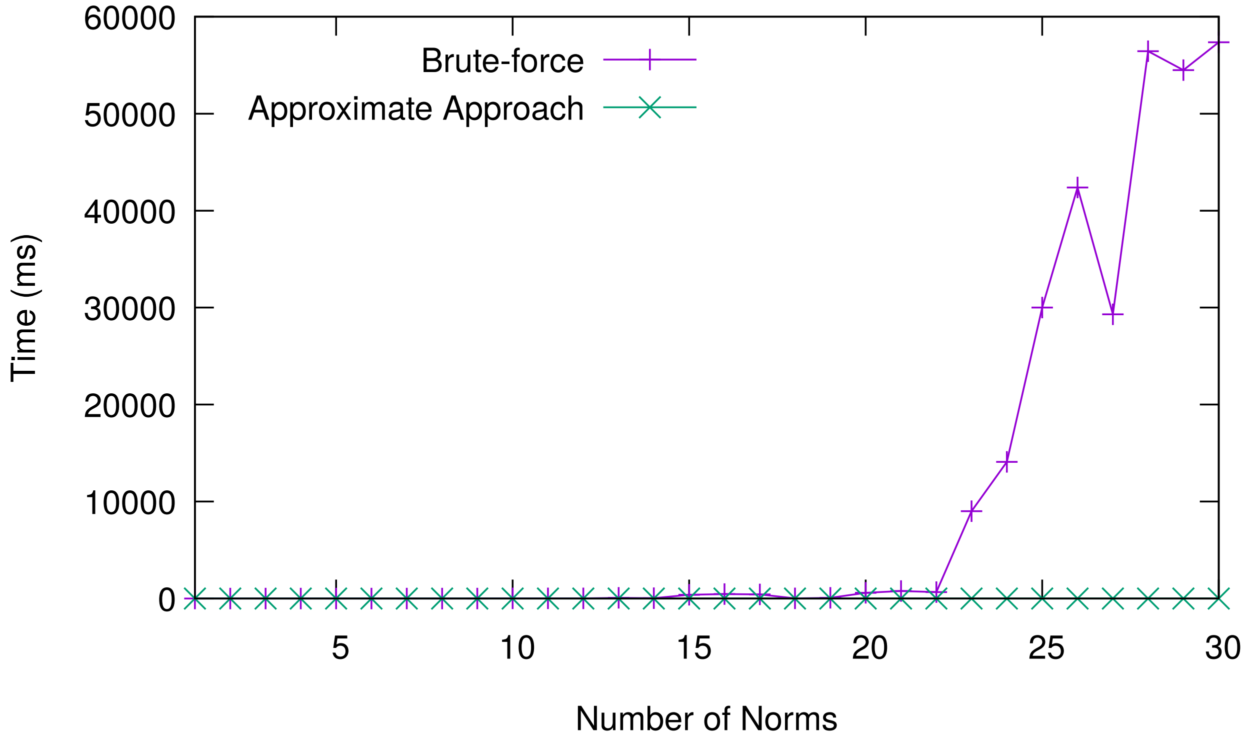

Figure 5.1 – Time Efficiency of Brute-Force and Approximate Solution, with timeout of 60 seconds

Figure 5.2 shows the time performance of both greedy solutions. The *Add and Update Solution* is worse in this metric than the *Norm Independence Solution* because it needs to recompute the *current cost* at each iteration. It is still fast compared to the brute-

[1]Open JDK VM build 25.151-b12, mixed mode

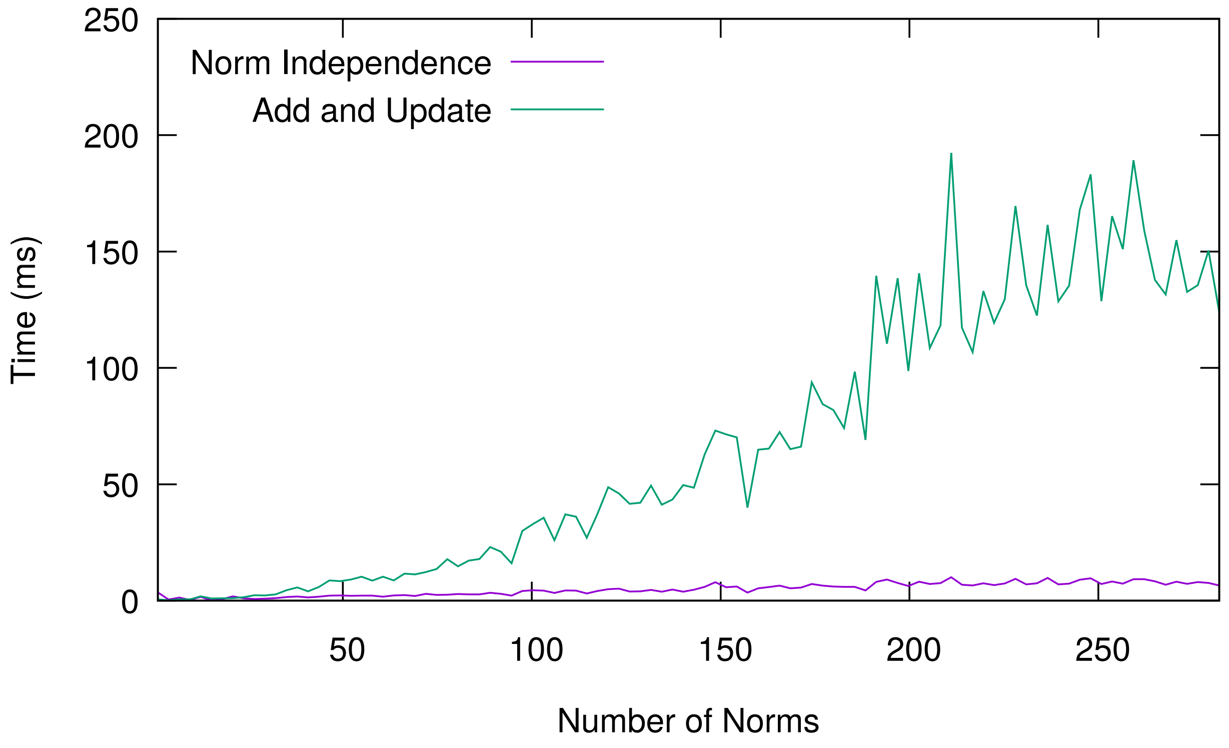

Figure 5.2 – Time Efficiency of both greedy solutions, smoothed using a sample of 100 data points and interpolated using splines

force approach, being able to find a solution for a problem with almost 300 norms in under one second.

We performed two experiments to compare the accuracy of the results of the approximate approaches. First, in Figure 5.4, we show the relative accuracy compared with an optimal solution to the problem using the brute-force approach. Note that, as we are comparing with the brute-force approach, these results are limited to small problems that this approach can solve[2]. In this experiment, both greedy approaches outperform the naive solution; between the two greedy approaches, the second one (*Add and Update Heuristic*) has, in almost all domains, greater accuracy, and — in all cases — a smaller standard deviation. For the *depots*, *gripper* and *logistics* domain, the second solution achieves optimal accuracy; this can be explained as these domains becomes intractable for really small number of norms, thus the number of problems in this experiment, for these domains, is also small. The increase in performance from the first to the second greedy approach is relatively small for these domains; while it is relatively large for the *drinkdriving* and *rooby* domains.

To investigate if the relations found for the first experiment hold for large problems, we perform additional experiments, shown in Figure 5.3. As these problems cannot be solved in a timely manner using a brute-force approach, in order to calculate their accuracy we compare them with a perfect solution that could monitor all norms, but that does not necessarily need to respect the available budget. This perfect solution has the maximum

[2]We set a timeout of 30 seconds for this experiment.

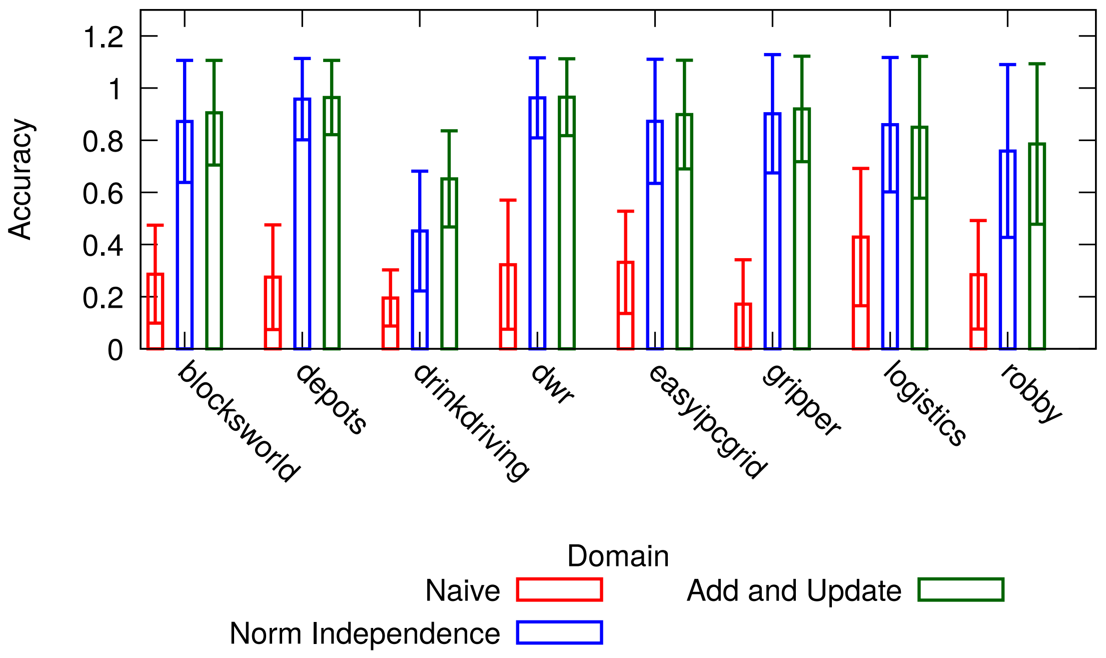

Figure 5.3 – Accuracy of approximate approaches, compared to the maximum U possible; error bars represent one standard deviation of uncertainty

value of **U**, which can be unattainable for actual solutions to these problems; therefore, in this experiment we are interested in the relative accuracy between the approximate approaches, and not in how close they are from the perfect solution.

We can see the same pattern in this experiment; the greedy approaches perform better than the naive solution, with a slight advantage for the *Add and Update Heuristic*. The increase in performance from the first to the second greedy approach remains large for the *drinkdriving* domain, while for other domains it is relatively small.

From the experiments we conclude that brute-force solution is intractable for all but small problems, while approximate approaches can solve large problems. The accuracy of the greedy approaches is better than the naive solution, with a slight advantage to the second greedy approach. This advantage is more noticeable for small problems; for large problems, as we do not have the value for the optimal solution, this difference is smaller.

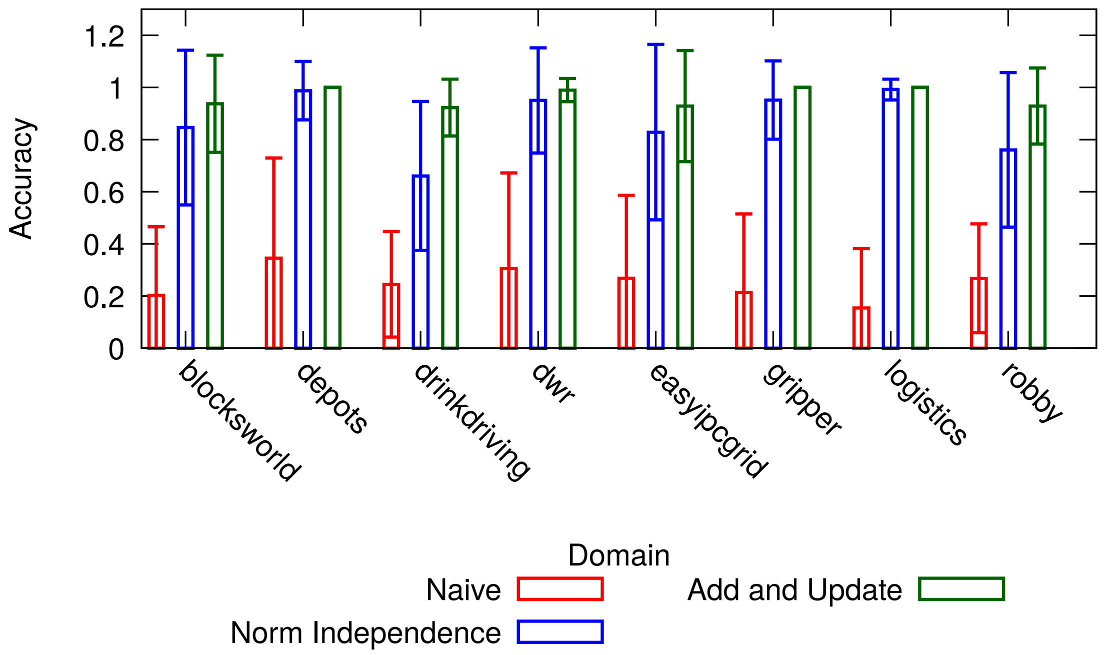

Figure 5.4 – Accuracy of approximate approaches, compared to a brute-force solution; error bars represent one standard deviation of uncertainty

6. CONCLUSION AND FUTURE WORK

In this work we report on our extension of the state of the art in the theory of norm monitoring by dropping the assumption that a monitor system has full observability, i.e. that monitors can observe all actions performed. Adding the notion of a set of available monitors and an associated cost results in the problem of finding a monitor placement in order to maximize the number of detectable violations. While brute-force solutions are impractical because the possible solution search space is exponential on the size of the input, we propose quasi-linear algorithms that use a mapping between norms and monitors to find approximate solutions. Our empirical of runtime performance and accuracy shows that these algorithms are both practical in computational terms and approach optimal performance for many realistic domains from the planning literature.

This work was published in the International Workshop on Linked Democracy: Artificial Intelligence for Democratic Innovation at IJCAI [KOM17]. We aim to extend the work in at least two ways. The first extension would be to consider different agents being able to perform concurrent actions, and how to build monitors able to correctly identify which agent violated a given norm. The second extension is to allow more complex expressions representing both what monitors can observe and how monitors can be combined, as currently we only consider conjunctions of monitors. This can increase the richness of our approach, and we intend to investigate heuristics for the use of such more expressive monitors.

REFERENCES

[ABDL15] Alechina, N.; Bulling, N.; Dastani, M.; Logan, B. “Practical run-time norm enforcement with bounded lookahead”. In: Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems, 2015, pp. 443–451.

[ADL14] Alechina, N.; Dastani, M.; Logan, B. “Norm approximation for imperfect monitors”. In: Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems, 2014, pp. 117–124.

[Bac01] Bacchus, F. “Aips 2000 planning competition: The fifth international conference on artificial intelligence planning and scheduling systems”, *AI magazine*, vol. 22–3, Jan 2001, pp. 1–47.

[BDK13] Bulling, N.; Dastani, M.; Knobbout, M. “Monitoring norm violations in multi-agent systems”. In: Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems, 2013, pp. 491–498.

[BDVP13] Balke, T.; De Vos, M.; Padget, J. “Evaluating the cost of enforcement by agent-based simulation: A wireless mobile grid example”. In: International Conference on Principles and Practice of Multiagent Systems, 2013, pp. 21–36.

[BVDTV06] Boella, G.; Van Der Torre, L.; Verhagen, H. “Introduction to normative multiagent systems”, *Computational & Mathematical Organization Theory*, vol. 12–2-3, Oct 2006, pp. 71–79.

[CCP$^+$98] Castelfranchi, C.; Conte, R.; Paolucci, M.; et al.. “Normative reputation and the costs of compliance”, *Journal of Artificial Societies and Social Simulation*, vol. 1–3, Jun 1998, pp. 1–3.

[CS17] Criado, N.; Such, J. M. “Norm monitoring under partial action observability”, *IEEE transactions on cybernetics*, vol. 47–2, Jun 2017, pp. 270–282.

[DGMT09] Dastani, M.; Grossi, D.; Meyer, J.-J. C.; Tinnemeier, N. “Normative multi-agent programs and their logics”. In: *Knowledge Representation for Agents and Multiagent Systems*, Springer, 2009, pp. 16–31.

[Dig99] Dignum, F. “Autonomous agents with norms”, *Artificial Intelligence and Law*, vol. 7–1, Jul 1999, pp. 69–79.

[DMSC00] Dignum, F.; Morley, D.; Sonenberg, E. A.; Cavedon, L. “Towards socially sophisticated bdi agents”. In: Proceedings of the Fourth International Conference on Multiagent Systems, 2000, pp. 111–118.

[EH21] Edelkamp, S.; Hoffmann, J. “Pddl2. 2: The language for the classical part of the 4th international planning competition”, Technical Report, University of Freiburg, 2004, pp. 21.

[EMWG10] El Mouden, C.; West, S. A.; Gardner, A. “The enforcement of cooperation by policing”, *Evolution*, vol. 64–7, Feb 2010, pp. 2139–2152.

[Eps01] Epstein, J. M. “Learning to be thoughtless: Social norms and individual computation”, *Computational economics*, vol. 18–1, Aug 2001, pp. 9–24.

[ERRAA04] Esteva, M.; Rosell, B.; Rodriguez-Aguilar, J. A.; Arcos, J. L. “Ameli: An agent-based middleware for electronic institutions”. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004, pp. 236–243.

[FvSM06] Fix, J.; von Scheve, C.; Moldt, D. “Emotion-based norm enforcement and maintenance in multi-agent systems: foundations and petri net modeling”. In: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, 2006, pp. 105–107.

[GCNRA05] García-Camino, A.; Noriega, P.; Rodríguez-Aguilar, J. A. “Implementing norms in electronic institutions”. In: Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multiagent Systems, 2005, pp. 667–673.

[GCRASV06] García-Camino, A.; Rodríguez-Aguilar, J.-A.; Sierra, C.; Vasconcelos, W. “Norm-oriented programming of electronic institutions”. In: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, 2006, pp. 670–672.

[GCRASV09] García-Camino, A.; Rodríguez-Aguilar, J. A.; Sierra, C.; Vasconcelos, W. “Constraint rule-based programming of norms for electronic institutions”, *Autonomous Agents and Multiagent Systems*, vol. 18–1, Feb 2009, pp. 186–217.

[GNT35] Ghallab, M.; Nau, D.; Traverso, P. “Automated planning: theory & practice”. Elsevier, 2004, pp. 635.

[JS93] Jones, A. J.; Sergot, M. “On the characterisation of law and computer systems: The normative systems perspective”. 1993, chap. 12, pp. 275–307.

[KN02] Kollingbaum, M. J.; Norman, T. J. “Supervised interaction: creating a web of trust for contracting agents in electronic environments”. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems, 2002, pp. 272–279.

[KOM17] Krzisch, G.; Oren, N.; Meneguzzi, F. “Bounded-monitor placement in normative environments”. In: Proceedings of the Workshop on Linked Democracy, 2017, pp. 28–37.

[Ld+02] Luck, M.; d’Inverno, M.; et al.. “Constraining autonomy through norms”. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems, 2002, pp. 674–681.

[LF03] Long, D.; Fox, M. “The 3rd international planning competition: Results and analysis”, *Journal of Artificial Intelligence Research*, vol. 20, Jun 2003, pp. 1–59.

[LMM+13] Luck, M.; Mahmoud, S.; Meneguzzi, F.; Kollingbaum, M.; Norman, T. J.; Criado, N.; Fagundes, M. S. “Normative agents”. In: *Agreement technologies*, Springer, 2013, pp. 209–220.

[MDS15] Meneguzzi, F.; De Silva, L. “Planning in bdi agents: a survey of the integration of planning algorithms and agent reasoning”, *The Knowledge Engineering Review*, vol. 30–1, Jan 2015, pp. 1–44.

[MFM+09] Modgil, S.; Faci, N.; Meneguzzi, F.; Oren, N.; Miles, S.; Luck, M. “A framework for monitoring agent-based normative systems”. In: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems, 2009, pp. 153–160.

[MMH02] Mui, L.; Mohtashemi, M.; Halberstadt, A. “Notions of reputation in multi-agents systems: a review”. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems, 2002, pp. 280–287.

[MMO+09] Meneguzzi, F.; Modgil, S.; Oren, N.; Miles, S.; Luck, M.; Faci, N.; Holt, C.; Smith, M. “Monitoring and explanation of contract execution: A case study in the aerospace domain”. In: Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems, 2009, pp. 153–160.

[RSGJ03] Ramchurn, S.; Sierra, C.; Godó, L.; Jennings, N. R. “A computational trust model for multi-agent interactions based on confidence and reputation”. In: Proceedings of the 6th International Workshop of Deception, Fraud and Trust in Agent Societies, 2003, pp. 69–75.

[SA85] Sutinen, J. G.; Andersen, P. “The economics of fisheries law enforcement”, *Land economics*, vol. 61–4, Jan 1985, pp. 387–397.

[SC11] Savarimuthu, B. T. R.; Cranefield, S. “Norm creation, spreading and emergence: A survey of simulation models of norms in multi-agent systems”, *Multiagent and Grid Systems*, vol. 7–1, May 2011, pp. 21–54.

[SP$^+$01] Staller, A.; Petta, P.; et al.. “Introducing emotions into the computational study of social norms: A first evaluation”, *Journal of Artificial Societies and Social Simulation*, vol. 4–1, Feb 2001, pp. U27–U60.

[SS07] Sierra, A. P. d. P. C.; Schorlemmer, M. “Friends no more: Norm enforcement in multi-agent systems”. In: Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems, 2007, pp. 92–94.

[Sti74] Stigler, G. J. “The optimum enforcement of laws”, vol. 3, Jan 1974, pp. 55–67.

[TDB16] Testerink, B.; Dastani, M.; Bulling, N. “Distributed controllers for norm enforcement”. In: Proceedings of the 22th European Conference on Artificial Intelligence, 2016, pp. 751–759.

[VdHW08] Van der Hoek, W.; Wooldridge, M. “Multi-agent systems”, *Foundations of Artificial Intelligence*, vol. 3, Oct 2008, pp. 887–928.

[VKN09] Vasconcelos, W. W.; Kollingbaum, M. J.; Norman, T. J. “Normative conflict resolution in multi-agent systems”, *Autonomous Agents and Multiagent Systems*, vol. 19–2, Nov 2009, pp. 124–152.

[yLLd06] y López, F. L.; Luck, M.; d’Inverno, M. “A normative framework for agent-based systems”, *Computational & Mathematical Organization Theory*, vol. 12–2-3, Oct 2006, pp. 227–250.