Planning under Uncertainty for Coordinating Infrastructural Maintenance[*]

Joris Scharpff Matthijs T.J. Spaan Leentje Volker Mathijs de Weerdt
Delft University of Technology, Delft, The Netherlands
{j.c.d.scharpff, m.t.j.spaan, l.volker, m.m.deweerdt}@tudelft.nl

ABSTRACT

We address efficient planning of maintenance activities on infrastructural networks, inspired by the real-world problem of servicing a highway network. A road authority is responsible for the quality, throughput and maintenance costs of the network, while the actual maintenance is performed by autonomous, third-party contractors.

From a (multi-agent) planning and scheduling perspective, many interesting challenges can be identified. First, planned maintenance activities might have an uncertain duration due to unexpected delays. Second, since maintenance activities influence the traffic flow in the network, careful coordination of the planned activities is required in order to minimise their impact on the network throughput. Third, as we are dealing with selfish agents in a private-values setting, the road authority faces an incentive design problem to truthfully elicit agent costs, complicated by the fact that it needs to balance multiple objectives.

The main contributions of this work are: 1) implicit multi-agent coordination on a network level through a novel combination of planning under uncertainty and dynamic mechanism design, applied to real-world problems, 2) accurate modelling and solving of the maintenance planning problem and 3) empirical exploration of the complexities that arise in these problems. Using two real-world application examples we introduce a formal model of the problem domain, present experimental insights and identify open challenges for both the planning and scheduling as well as the mechanism design communities.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Algorithms, Experimentation

Keywords

Multiagent systems, Planning and Scheduling, Coordination

1. INTRODUCTION

The planning and scheduling of maintenance activities on large infrastructural networks, such as a national highway network, is a challenging real-world problem. While improving the quality of the infrastructure, maintenance causes temporary capacity reductions of the network. Given the huge impact of time lost in traffic on the economic output of a society, planning maintenance activities in a way that minimises the disruption of traffic flows is an important challenge for the planning and scheduling field. In this paper, we address this challenge by a novel combination of stochastic multi-agent planning, captured in Markov Decision Processes (MDPs), and dynamic mechanism design.

Maintenance activities in infrastructural networks not only incur maintenance costs, but also incur *social costs* based on the effects maintenance has on the throughput of the network. Put simply, closing off a main highway will lead to traffic delays, resulting in quantifiable losses in economic activity. For instance, an hour of time lost in traffic is valued between €10 for leisure traffic and €40 for business traffic [22]. Since maintenance activities are unavoidable, traffic delays are inevitable. However, they can be *minimised* using planning and scheduling techniques. In our work, we use planning-under-uncertainty methodology to schedule maintenance activities across the entire network, taking into account the effect that road closures have on traffic on nearby road segments.

A powerful real-world example of the benefits that careful maintenance planning can provide is the summer 2012 closure of the A40 highway in Essen, Germany. Instead of choosing for the default option of restricting traffic to fewer lanes for 2 years, authorities fully closed off a road segment for 3 months and diverted traffic to parallel highways. Traffic conditions on the other highways hardly worsened, while an estimated €3.5M in social costs due to traffic jams were avoided (besides lowering building costs) [9].

As maintenance activities often have an uncertain duration due to delays in construction, it is important to take uncertainty into account while planning. Also, there may be multiple ways to perform a certain maintenance action by varying the amount of resources dedicated to it, leading to options that have different duration, cost, risk and effect on asset quality. Furthermore, long-term planning is required to ensure overall network quality. Markov Decision Processes provide a suitable framework to model and solve these types of planning-under-uncertainty problems [19].

A complicating factor, however, is that while a single public road authority is responsible for the quality, throughput and costs of the network, the actual maintenance is performed by autonomous contractors, typically third-party companies interested primarily in maximising their profits. Road authorities face the problem of aligning objectives; we introduce monetary incentives for the contractors to consider global objectives. But an agent servicing one part of the network also influences agents in other parts as his work has a negative impact on the traffic flow. As a consequence, the payments we introduce lead to very high throughput penalties for all agents if

[*]This work will also appear in the Proceedings of the ICAPS 2013 conference [20].

Appears in *The Eighth Annual Workshop on Multiagent Sequential Decision-Making Under Uncertainty* (*MSDM-2013*), held in conjunction with *AAMAS*, May 2013, St. Paul, Minnesota, USA.

they do not coordinate their maintenance plans on a network level.

In this work we focus on socially optimal joint maintenance planning that maximises the sum of contractor utilities, in the presence of such monetary incentives, and therefore we have chosen a centralised coordination approach. The authority is given the responsibility to develop socially optimal plans, while considering the individual interests of all contractors expressed through cost functions. However, as these cost functions are private information, optimal coordination and hence outcomes can only be achieved if the contractors report these costs *truthfully*. Ensuring this truthfulness is the key motivation to combine stochastic planning with mechanism design.

Our main contribution is the application of a combination of stochastic planning and dynamic mechanism design to realise truthful coordination of autonomous contractors in a private-values setting. Typical one-shot mechanisms often used to elicit private values are not suitable for contingent and repeated settings. Instead we focus on *dynamic mechanisms* that define payments over all expected outcomes such that in expectation it is in the agent's best interest to be truthful during the entire plan period. Applying dynamic mechanism design to (real-world) settings such as the one we consider is relatively unexplored territory [6].

Related Work

Other approaches towards solving the problems discussed here have been considered, although they can not be applied to our setting for various reasons. Multi-agent MDP [4] assumes co-operative agents that are willing to share private information and have the same utility functions. This is not the case with decentralised MDP [3], however agent decisions are made solely on locally available information and are therefore inadequate in optimising network objectives. Moreover, both methods are not suitable when agents misreport their private information to 'cheat' the center into different outcomes. Non-cooperative settings have been studied in the classical planning literature [5, 14, 23], but uncertainty is not addressed.

Multi-machine scheduling has also been considered for the planning of maintenance activities, but we found this infeasible for our contingent setting. Moreover, finding optimal policies in multi-machine problems with general cost functions is highly intractable. The only work we are aware of in this area is [10], in which only non-decreasing regular step functions are considered. In our problem agents could both profit as well as suffer from concurrent maintenance, therefore our cost functions do not have the non-decreasing property.

Another interesting related approach is that of reinforcement learning [15, 16] and in particular Collective Intelligence [25]. In this an approach agents learn how and when to coordinate and, in the case of collective intelligence, strive to optimise a global goal, without substantial knowledge of the domain model. Nevertheless, the absence of domain knowledge makes it impossible for such methods to provide theoretical guarantees regarding agent and system performance – crucial if this method is to be applied in practice (e.g. as part of a dynamic contracting procedure) – and will most likely not produce socially optimal solutions.

Applying mechanism design to large multi-agent systems is challenging. First, few results are known for dynamic settings (but see [21]), but taking into account changes in the state of the system is crucial for planning ahead. Second, for large systems, computing optimal solutions might not be possible. When resorting to approximate solutions, however, standard theory for strategy-proof mechanisms does not immediately apply [18]. Finally, existing general solutions fail to apply in settings with actuation uncertainty. For instance, fault-tolerant mechanism design has been proposed to take into account that agents might not be able to accomplish an assigned activity [17], but no techniques exist for the quite general forms of uncertainty that we address.

Both stochastic planning and mechanism design have been well studied independently, however only a handful of papers address dynamic mechanism design and/or a combination of the two. [2] proposed a dynamic variant of the VCG mechanism for repeated allocation, implementing the mechanism desiderata in a within-period, ex-post Nash equilibrium. [1] studied a dynamic variant of the AGV mechanism [8] that is budget-balanced in the weaker Bayes-Nash equilibrium solution concept. Highly related is the work by [7], in which the authors also study dynamic mechanism design to obtain desirable outcomes in multi-agent planning with private valuations. However, the focus is on allocation problems that can be modelled as multi-armed bandit problems, instead of the richer problem domains with dynamic states that we consider.

Outline

In the next section, we present a theoretical framework for maintenance planning obtained and refined through interviews and discussions with public road and rail network authorities, as well as asset managers of several larger contractors. We then introduce the theoretical background of both stochastic planning and mechanism design (Section 3), and show how to combine work on planning with uncertainty and dynamic mechanism design to solve two example applications, derived from practice (in Section 4). We present experimental insights where we compare this with uncoordinated agents as well as with best-response (Section 5). In our conclusion, we summarise our findings and present open challenges for both the planning and scheduling as well as the mechanism design communities (Section 6).

2. MAINTENANCE PLANNING

Commonly in infrastructural maintenance planning there is one (public) institution responsible for the network on behalf of the network users. This road authority is given the task to maintain a high (i) network quality and (ii) throughput (iii) at low costs (although other objectives are also possible, e.g. environmental concerns, robustness, etc.). To this end, network maintenance has to be performed with minimal nuisance. However, the actual maintenance is performed by several autonomous, independent contractors and therefore some coordination of maintenance activities is required.

The maintenance planning problem discussed in this paper is part of the dynamic contracting procedure introduced in [24]. Here we focus on the execution phase, i.e. the planning and execution of maintenance activities. The assessment, pricing and allocation of maintenance activities is performed in the preceding procurement phase, not discussed in this paper.

In the infrastructural maintenance planning problem we are given a network of roads E. On this network we have a set N of agents (the contractors), with each agent $i \in N$ responsible for the maintenance of a disjoint subset $E_i \subseteq E$ of roads over a set of discrete periods T. An edge $e_k \in E$ has a quality level $q_{e_k} \in [0, 1]$ and a function $\hat{q} : q \times T \rightarrow q$ that models the quality degradation of a road given the current state and time (new roads degrade less quickly, seasons influence degradation).

For each edge $e_k \in E_i$, an agent i has a set of possible maintenance activities A_k that have been identified and assigned in the aforementioned procurement phase. We write A_i to denote all possible activities by an agent i, i.e. $A_i = \cup_{\{k|e_k \in E_i\}} A_k$. Each of the activities $k \in A_k$ has a duration $d_k \in \mathbb{Z}^+$, a quality impact function $\Delta q_k : q_{e_k} \times T \rightarrow q_{e_k}$ that depends on the current road quality and time, and a constant revenue $w_k \in \mathbb{R}$ that is obtained upon completion of the activity. Moreover, the agent has a (private) cost function $c_i : A_i \times T \times \mathbb{Z}^+ \rightarrow \mathbb{R}$ that represents the cost of performing an activity $k \in A_i$ at time $t \in T$ for a duration $d \in \mathbb{Z}^+$. The dependency on time enables modelling of different costs for example for different seasons, or for periods where the agent has fewer resources available.

We model the limited resources (machinery, employees, etc.) available to an agent by allowing at most one activity at a time. This restriction does not have much impact on the model we propose here but does greatly simplify resource reasoning and therefore the complexity of finding optimal maintenance plans.

Each agent strives to plan their maintenance activities in such a way that their profits are maximised, but plan execution is unlikely to be perfect. Uncertainties in various forms – for example delays, unknown asset states, failures, etc. – may be encountered during execution and hence 'offline optimal' plans might lead to rather poor results. To this end we focus on contingent plans, or *policies*, that dictate the best action to take *in expectation* for all possible agent states. Note that actions here are operations available to the contractors (e.g. start activity, do nothing, etc.) and states contain all relevant information for its planning problem. We formalise all these concepts in Section 3.1, for now it is sufficient to know that we can always obtain contractor plans by evaluating the most-likely path of a policy (i.e. the sequence of actions that in expectation optimise the contractor reward). We use $(k, t) \in \pi_i$ to denote that from the policy π_i we can derive that starting activity k at time t is expected to yield the highest reward.

Given a policy π_i, the expected profit for agent i is defined as

$$C_i(\pi_i) = \sum_{(k,t)\in\pi_i} \bar{w}_k - \sum_{(k,t)\in\pi_i} c_i(k, t, d_k) \qquad (1)$$

with $\bar{w}_k$ being w_k if the activity is completed within the period T and 0 otherwise. As activity rewards follow directly from the procurement, we assume that agents in expectation are always able to achieve a positive profit for completing their activities, otherwise they would not have bid on the activity during procurement. Note that we do not explicitly require all activities of an agent to be planned or that they can be completed within the period T, but because agents will not receive rewards w_k for each uncompleted activity k they will be stimulated to complete them.

For the agents to also consider the global objectives, we introduce payments such that their profits depend on the delivered quality and additional congestion caused by their presence. The quality payment Q_i for each agent i can be both a reward as well as a penalty, depending on the final quality state of its roads (e.g. based on contracted demands). Again given a policy π_i, we can determine the resulting quality state q_e^T at the end of the period T using the recursive formula

$$q_{e_k}^{t+1} = \begin{cases} \Delta q_k(q_{e_k}^t, t) & \text{if } (k, t) \in \pi_i \\ \hat{q}_{e_k}(q_{e_k}^t, t) & \text{otherwise} \end{cases} \qquad (2)$$

with (given) initial quality $q_{e_k}^0$. Consequentially, we define the quality payment for agent i as a result of his policy by $Q_i(\pi_i) = \sum_{e\in E_i} Q_i(q_e^T)$.

Congestion payments, i.e. social costs, cannot be considered from just the single agent perspective because network throughput depends on the planning choices of all agents. Let A^t denote the set of activities performed by all agents at time t, then the social cost of this combination is captured by $\ell(A^t)$. The impact of an individual agent, given the choices made by others, can be determined by $\ell_i(A^t) = \ell(A^t) - \ell(A^t_{-i})$ in which A^t_{-i} denotes the set of activities performed at time t minus any activity by agent i. The social cost function can for example capture the costs of traffic jams due to maintenance activities. In realistic scenarios where a good estimate of origins and destinations of the traffic flow is known, these costs can be learned empirically.

Recapitulating the above, each agent i is interested in maximising its expected profit, trivially comprised of only maintenance rewards and costs C_i. In order to stimulate agents to plan maintenance in favour of global objectives, we introduce quality and throughput payments such that their profit u_i, given the *joint* policy $\pi = \bigcup_{i\in N} \pi_i$, is now given by:

$$u_i(\pi) = C_i(\pi_i) + Q_i(\pi_i) + \ell_i(\pi) \qquad (3)$$

in which $\ell_i(\pi) = \sum_{t\in T} \ell_i(A^t)$. Here, A^t can be easily derived from π considering planned start times and activity durations as before.

Now given the utility function of Eq. 3, how should we define these payments such that the right balance is made between these costs and the agents' private costs, which are not known to the road authority? This is exactly a mechanism design problem. As the scenarios both contain a form of uncertainty and thus dynamics, there are few known mechanisms that can be applied; we are aware of only two (dynamic-VCG [2, 6] and dynamic AGV [1]). Both require an optimal solution (in expectation) of the planning problem, otherwise agents can benefit from (deliberately) misreporting their private costs.

In the next section we therefore start by discussing how to compute optimal solutions to the problem variants introduced in this section, followed by a summary of how this can be combined with a dynamic mechanism.

3. BACKGROUND

We briefly introduce the two concepts our work builds on, planning under uncertainty and dynamic mechanism design.

3.1 Planning under Uncertainty

To deal with uncertainties we model the planning problem using Markov Decision Processes (MDPs), which capture this type of uncertainty rather naturally [19]. For each agent $i \in N$ we have an MDP $M_i = \langle S_i, \mathcal{A}_i, \tau_i, r_i \rangle$ that defines its local planning problem. In this definition, S_i is the set of states and $\mathcal{A}_i$ a set of available actions (see Sections 4.3 and 4.4). The function $\tau_i : S_i \times A_i \rightarrow \Delta(S_i)$ describes the transition probabilities where $\tau_i(s_i, \mathcal{A}_i, s_i')$ denotes the probability of transitioning to state s' if the current state is s_i and action $\mathcal{A}_i$ is taken. Finally, $r_i : S_i \times \mathcal{A}_i \rightarrow \mathbb{R}$ is the reward function where $r_i(s_i, a)$ denotes the reward that the agent will receive when action $a \in \mathcal{A}_i$ is taken in state s_i (e.g. the utility of Eq. 3). We formalise the rewards and actions for the agents in Section 4, as they depend on the encoding used to solve the MDP.

Solutions to MDPs are policies $\pi : S \rightarrow \mathcal{A}$ that dictate the best action to take in expectation, given the current state it is in. Formally, the optimal policy π^* is defined such that for all start states $s \in S$:

$$\pi^*(s) = \underset{\pi\in\Pi}{\arg\max}\ V^0(\pi, s) \qquad (4)$$

with

$$V^{t_0}(\pi, s) = \mathbb{E}\Big[\sum_{t=t_0}^{\infty} \gamma^t r(s^t, \pi(s^t))) \mid s^{t_0} = s\Big] \qquad (5)$$

in which s^t is the state at time t and $\gamma \in [0, 1)$ is a shared discount factor commonly used to solve problems with infinite horizons.

We can obtain the individual policies π_i for each agent by solving its local plan problem MDP M_i. However, in order to develop an (optimal) joint policy π^*, required to consider throughput payments, we need to solve the multi-agent MDP that results from combining all individual MDPs. Formally, the joint MDP is defined by $M = \langle S, \mathcal{A}, r, \tau \rangle$ where $S = \times_{i \in N} S_i$ is the joint state space containing in each state $s \in S$ a local state s_i for all agents $i \in N$, $\mathcal{A}$ is the set of combined actions, r the reward function defined as $\forall s \in S, a \in \mathcal{A} : r(s,a) = \sum_{i \in N} r_i(s_i, a_i)$ and τ the combined transition probability function. The joint action set can always be obtained by including an action for each element of the Cartesian product set of all individual action spaces but smarter construction can greatly reduce the joint action set. For planning problems (at least) we have developed a two-stage MDP encoding that effectively reduces the joint action set size from exponential to linear in the number of players and their action sets. This is discussed in detail in Section 4.2.

3.2 Dynamic mechanism design

Although MDPs facilitate modelling uncertainties, they assume global knowledge of all these uncertainties, as well as costs and rewards. As the maintenance activities are performed by different, usually competing companies, we cannot assume that this knowledge is globally available. We therefore aim to design a game such that utility-maximising companies behave in a way that (also) maximises the global reward. This is exactly the field of mechanism design, sometimes referred to as inverse game theory.

Formally, in a static or one-shot game, each agent $i \in N$ has some private information θ_i usually known as its *type*. In so-called direct mechanisms, players are asked for their type, and then a decision is made based on this elicited information. Groves mechanisms [12] take the optimal decision (π^*) and define payments $\mathcal{T}$ such that each player's utility is maximised when it declares its type truthfully.

Dynamic mechanisms extend 'static' mechanisms to deal with games in which the outcome of actions is uncertain and private information of players may evolve over time. In each time step t, players need to determine the best action (in expectation) to take while considering current private information and possible future outcomes. Private rewards are therefore defined depending on the state and the policy, given by $r_i(s^t, \pi(s^t)))$, in which the state contains the player's type. This type is denoted by θ_i^t to express the possibility of this changing over time. With θ^t we denote the type of all players at time t which are encoded in the state s^t.

An extension of Groves mechanisms for such a dynamic and uncertain setting is dynamic-VCG [2, 6]. For dynamic-VCG the decision policy is the optimal one maximising the reward of all players, identical to Eq. 4 when the types θ^t are encoded into the state s^t. We denote this optimal policy for time step t given the reported types θ^t encoded in state s^t by $\pi^*(s^t)$. A policy optimised for the game with all players except i is denoted by $\pi^*_{-i}(s^t)$ and we define $r_i(s_i^t, \pi^*_{-i}(s_i^t)) = 0$.

In every time step each player i pays the expected marginal cost he incurs on other players j for the current time step. This is defined as the difference between the reward of the other players for the socially optimal decision for the current time step t, i.e. $\sum_{j \neq i} r_j(s^t, \pi^*(s^t))$ and their expected reward optimised for just them in future time steps, i.e. $V^{t+1}(\pi^*_{-i}, s^{t+1})$ (Eq. 5) minus the expected reward of the other players for a policy optimised for them for all time steps including the current one, i.e. $V^t(\pi^*_{-i}, s^t)$. Summarising, the payment $T_i(\theta^t)$ for an agent i at time step t given that reports θ^t are encoded in state s^t is thus is defined as

$$T_i(\theta^t) = \sum_{j \neq i} r_j(s^t, \pi^*(s^t)) + V^{t+1}(\pi^*_{-i}, s^{t+1}) - V^t(\pi^*_{-i}, s^t) \tag{6}$$

The dynamic-VCG mechanism yields maximum revenue among all mechanisms that satisfy efficiency, incentive compatibility and individual rationality in within-period, ex-post Nash equilibrium. This means that at all times for each player the sum of its expected reward and its expected payments is never more than when declaring its true type.

4. COORDINATING MAINTENANCE PLANNING

In this work we combine existing work on planning under uncertainty and dynamic mechanism design to solve the complex problem of maintenance planning where agents are selfish and execution is uncertain. Using the dynamic VCG mechanism we ensure that agents are truthful in reporting their costs. Then, using these reports to model agent rewards, we apply planning-under-uncertainty techniques to find efficient policies and finally we determine the payments of the mechanism, as discussed in the previous section. Basically this approach can be seen as a series of auctions in which agents bid activities and their associated costs for an assignment of time-slots.

An important condition for the dynamic VCG mechanism is that the chosen policy is optimal. If it is not, the payments are not guaranteed to achieve truthful cost reports and agents may want to deviate. Therefore we focus on exact solving methods in our approach.

We implemented our mechanism using the SPUDD solver [13] to determine optimal policies. The SPUDD solver allows for a very compact but expressive formulation of MDPs in terms of algebraic decision diagrams (ADDs) and uses a structured policy iteration algorithm to maximally exploit this structure. This allows it to find optimal solutions to moderately sized problems. We note, however, that our mechanism is independent of the particular MDP solver used, as long as it returns optimal solutions.

4.1 MDP models for maintenance planning

Recall that we want to find an efficient joint policy π^* that maximises the sum of all agent utilities u_i (Eq. 3). However, this utility function cannot be directly translated into an equivalent MDP encoding. Although in our model C, Q and ℓ can be general functions, encoding general functions in the MDP formulation potentially requires exponential space. Hence to be able to use the SPUDD solver in our experiments, we necessarily restricted ourselves to only linear functions.

The current state of the network, i.e. the quality levels q_e, are modelled using a 5 star classification (from (0) very bad to (5) excellent) are encoded as discrete variables $[0, 5]$. Road degradation functions $\hat{q}$ are modelled using decision diagrams that probabilistically decrease the road quality in each time slot by one state. Completing an activity k' increases the corresponding road quality q'_k by a specified number of states (additive), corresponding to its effect $\Delta q'_k$.

Encoding the social cost ℓ can be cumbersome, depending on the complexity of the chosen cost model. Again, general cost models could result in exponential MDP encoding sizes. Using only unary and binary rules to express social cost, we can overcome this exponential growth. (at the cost of losing some expressiveness). The unary rules $l : A \to \mathbb{R}$ express the marginal latency introduced by each activity independently. Dependencies between activities are expressed using binary relations $l : A_i \times A_j \to \mathbb{R}$ that specify the additional social cost when both activities are planned concurrently. The costs incurred by the set of chosen activities A^t can be computed using $\ell(A^t) = \sum_{k \in A^t} l(k) + \sum_{k_1 \in A^t} \sum_{k_2 \neq k_1 \in A^t} l(k_1, k_2)$.

4.2 Avoiding exponentially-sized action spaces

Factored MDP solvers are typically geared towards exploiting structure in transition and reward models, but scale linearly with the number of actions. In multi-agent problem domains such as ours, however, a naive construction of the joint action set – such as enumerating all elements of the Cartesian product of individual action sets – can be exponential in the number of agents. To overcome this issue, we model each time step in the real world by two stages in the multi-agent MDP, resulting in a larger number of search backups due to additional variables, but crucially avoiding exponentially-sized action spaces. Note that the encoding technique we discuss in this section is not restricted to our problem; they can be applied on any multi-agent decision problem MDP formulation in which agent actions are dependent only through their rewards.

In our MDP encoding we have used a two-stage approach for each time step in the plan problem length T. In the first step agents decide on the activity to perform (or continue) and this activity is then 'executed' in the second stage (illustrated in Sections 4.3 and 4.4 for two example scenarios). We implement this separation through the use of additional variables that for each agent state the activity to perform in the current time step. These variables can be set independent from the actions available to other players (unlike the Cartesian product action space). The second stage then encodes the 'execution' of their choices using one additional action. Still there are multiple ways in which this first-stage activity selection can be implemented. Again enumeration is possible (although obliterating the purpose of the two-stage approach) but we have developed two smarter encodings: action chains and activity chains.

The action chain encoding exploits the fact that we can decide on an action for each player sequentially, instead of deciding on them all at once (as with enumeration). Through the use of a player token, each agent gets a 'turn' to determine his action within a single time step. Therefore we require only $|A_i|$ actions for each agent i, one for each activity it can choose, and hence a total of $\sum_{i \in N} |A_i|$ states (and one additional variable), instead of the $\prod_{i \in N} |A_i|$ actions needed for enumerating the Cartesian product.

For activity chains we exploit a similar idea. We group the activities of agents into activity chains to obtain an even smaller set of joint MDP actions. Let $D = \max_{i \in N} |A_i|$ be the size of the largest activity set of any player, then the activity chains are defined as $AC_m = \bigcup_{i \in N} k_m \in A_i$ for $m = 1, 2, \ldots, D$. Hence we group all m-th activities of each player into set AC_m. If a player i has no m-th activity, i.e. $m > |A_i|$, we exclude the player from this activity chain using a high penalty. Again using the player token we enforce that each player sequentially chooses an activity from one of these chains. This encoding requires exactly D actions in the joint MDP for the first stage and is therefore often more compact than the $\sum_{i \in N} |A_i|$ actions required for action chains.

In the second stage we model the execution of these choices, i.e. apply maintenance effects, and compute the sum of utilities (Eq. 3) for this time step as the reward. Note that we only proceed in time after the second stage, hence both stages are effectively within one time slot $t \in T$.

So far we have introduced a general encoding for maintenance scheduling problems. Now we will go into the the specifics for two real-world application we have chosen to study in this paper: one with unit-time activities that may fail, and one where activities always succeed, but possibly have a much longer duration. A summary of the main differences can be found in Table 1.

	repeat	duration	success prob.	delay dur.	delay prob.
		d_k	α_k	h_k	β_k
1	yes	1	$[0, 1]$	0	0
2	no	$\mathbb{Z}^+$	1	$\mathbb{Z}^+$	$[0, 1]$

Table 1: The differences between scenario 1 and 2

4.3 Scenario 1: Activities with failures

As a step towards network maintenance, we first focus on scheduling repeatable unit-time activities with possible failures. Although this problem is conceptually rather simple, it captures essential parts of real-world applications such as factory scheduling and supply chain planning problems. In this scenario, activities $k \in A_i$ are repeatable, of unit-time ($d_k = 1$) and succeed with probability $\alpha_k \in [0, 1]$. It is possible for any activity $k \in A_i$ to fail with probability $1 - \alpha_k$. Whether an activity fails will become apparent at its actual execution time. When an activity fails, it has no positive effect on the quality but its associated maintenance and throughput costs are still charged. If the agent still wants to perform the maintenance it has to include the activity in his plan again at a later time.

Because activities in this scenario are unit-time and repeatable, we can directly translate these into actions of the single-agent MDPs. For each activity $k \in A_i$ of agent i we create an action a_k with reward $c(k, t, 1)$. This action improves the quality level q_k by the number of levels corresponding to Δq_k with probability α_k. Thus with probability $1 - \alpha_k$ the maintenance fails and the quality level remains unchanged.

4.4 Scenario 2: Portfolio Management

Portfolio management is a second variant of our model. Inspired by real-world consequences of signing a maintenance contract, in this setting agents have to perform each activity exactly once, although multiple alternatives exists for the activity, and instead of activity failure we consider delays. More formally, for each activity k we now additionally have a delay duration h_k and delay probability β_k.

Encoding the portfolio management planning in an MDP requires a substantially greater effort as we can no longer translate activities directly to actions. This problem is more complex because of (1) possible non-unit activity durations, (2) activities can be delayed, (3) for each road we can only choose one activity to perform, and (4) each road can be serviced only once. The latter two are easily resolved by introducing a variable that flags whether a road has been serviced and using corresponding penalties to prohibit planning of these activities in a later time; the first two require more work.

From the single agent MDP perspective, non-unit activity durations (including possible delay) do not pose any difficulties. We could use actions that update the time variable t according to the activity duration. For the joint MDP however, this time variable is shared over all the agents. Increasing the time by the activity duration makes it impossible for other agents to start their activities in this time period.

Our solution is to decompose each activity k into a series of unit-time MDP actions $\{start_k, do_k, delay_k, done_k\}$ and use a timer variable to keep track of the remaining activity duration and its delay status ($pending$, no or yes). The $start_k$ action marks the beginning of the activity. This action sets the delay status to $pending$ and the activity timer to the duration d_k. In subsequent time steps, the agent has to perform a do_k action until the activity timer reaches zero. At this point, the activity delay status is pending and the activity is delayed with probability β_k (also updating the delay status).

If the activity is not delayed, the $done_k$ action is executed and the associated road e_k is flagged as serviced. When an activity is delayed however, we set the activity timer to the delay duration h_k and continue with do_k actions until again the timer reaches zero, at which point the $stop_k$ action is executed (not $delay_k$ again because of the delay status value).

Important to keep in mind is that during the search for optimal policies, a solver might decide on any order of these actions. Hence we need to constrain the actions such that only feasible action sequences are considered. For example, the do_k action can only be chosen if the activity timer is greater than zero, otherwise a high penalty results.

Rewards are encoded using the two-stage approach as before. In the first stage, each agent chooses a $start$, do, $delay$ or $stop$ action. Then the second stage implements these actions and incurs maintenance, quality and social costs for the current time step t.

4.5 Planning Methods

Using the encodings we discussed, we can find the optimal policy π^* that minimises costs over all three objectives. In the experiments, we then compare this centralised computation that relies on truthful reporting to (1) the approach where each agent plans its own actions optimally individually, i.e. disregarding other agents, and (2) a best-response approach [14].

In the best-response approach, agents alternatingly compute their best plan (in expectation) in response to the current (joint) plan of the others. This approach allows us to solve much easier single agent problems but still consider agent dependencies (e.g. social cost). Of course, the downside of this approach is that we will have to settle for Nash equilibria (if they exist).

5. EVALUATION

The combination of planning under uncertainty and mechanism design has been studied in only a handful of papers, their focus mainly on theoretical aspects. Little is known about application of such a combination on practical problems. We have performed a substantial number of experiments to gain insight into this previously uncharted area.

For both problem scenarios we have generated large benchmark sets on which we tested the various planning approaches and their encodings discussed in the previous section. These experiments are mainly of an exploratory nature in which we study the effect of each of the problem variables.

The solver used in these experiments has been implemented in Java, using SPUDD as its internal MDP solver. All experiments have been run on a system with an 1.60 Ghz Intel i7 processor.

5.1 Activities with failures

In the first series of experiments we have been mainly interested in exploring the computational limits of solving the problem centrally using an exact algorithm. To this end we generated a set of simple instances that vary in both the number of players N (2-5) and activity set A_i sizes (1-15). We solved these instances using different planning period lengths T (1-46). From these experiments we identify the parameters that contribute the most to the complexity of the problem.

Activity sets are generated using random, linear, time-dependent cost functions and always increase the quality level of the associated road by one. Quality cost functions are also generated for each road. Road quality is decreasing linearly in the quality with a random factor from [1, 3], which is fixed per road. Recall from Section 4.1 that linearity of this and other cost functions is a restriction not imposed by our model but is required to combat a potential exponential MDP encoding size.

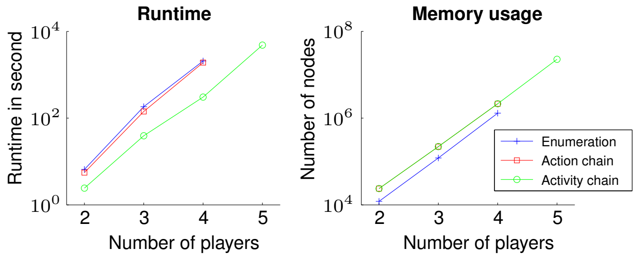

Figure 1: Comparison of runtime (left) and memory use (right) for different encoding methods and number of player, $|A_i| = 3$, $|T| = 46$, $|Q| = 6$ (both log scale).

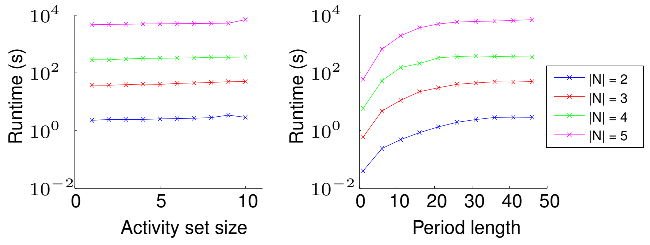

Figure 2: Runtime for different activity set sizes with $T = 46$ (left) and plan period lengths with $|A_i| = 10$ (right) using the activity chain encoding (again both log scale).

The choice of activity and quality cost functions mostly influence the rewards players can obtain, their impact on the computational complexity is negligible (unless very complex models are used to compute costs). Regarding the social costs ℓ we study the worst-case where all activities always interfere and define these costs using randomly chosen (marginal) cost $l(k1, k2) \in [1, 10]$ for each $k_1 \in A_i$ and $k_2 \in A_j$ where $i \neq j$. We do not consider the marginal cost for individual actions, i.e. $l(k) = 0$.

In Figure 1 we have depicted both the runtime (left) and the memory (right) required to solve each of these instances, under different encoding methods. The memory required is expressed in the number of nodes SPUDD generates.

Not surprisingly this figure illustrates that the performance of the solver is exponential in both time and memory, and greatly depends on the structure of its input. By exploiting the problem structure, the activity chain encoding is able to greatly reduce the required runtime. With it we have been able to solve instances with 5 players and 3 activities per player within the time limit of 3 hours, whereas the other two failed on such instances. Observe that activity chain encoding requires slightly more memory, but this will be of no burden to most modern day computer systems.

For the reasons stated above, we have illustrated the results of the remaining experiments only using the activity chain encoding (which indeed outperformed the others in all tests). In Figure 2 we have plotted the required runtime for solving instances using activity chains for various activity set sizes and period lengths.

From the figure we can conclude that the runtime is only linearly affected by the number of activities each player has. The plan period length shows almost the same: although the required runtime increases rapidly at first, for larger plan horizons the increase is again almost linear. It is expected that instances with small plan lengths are easily solvable because only a small number of plans

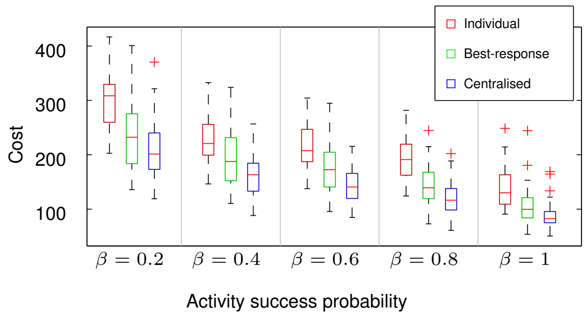

Figure 3: Total cost using different planning approaches for the activities with failure problems (lower is better).

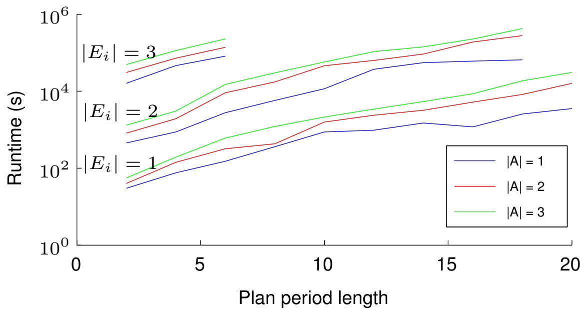

Figure 4: Runtimes of best-response planning for portfolio management for various road set sizes $|E_i|$, activities per road $|A|$ and plan length T (log scale).

is possible. Increasing the plan length introduces an exponential number of new possible plans and therefore the computation time increases rapidly, up to the point where the roads reach maximum quality. From this time on, agents have to consider planning an activity only when the quality degrades.

Having identified the computational boundaries of the centralised problem, we compared the performance of different planning approaches discussed in Section 4.5 in terms of total reward obtained. For these experiments we have used 60 generated two-player instances in which each player is responsible for one road.

The activity set of each player contains the no-operation and 1, 2 or 3 available maintenance operations that improve the quality of the road by 1, 2 or 3 levels respectively. The cost of each action $k \in A_i$ is drawn randomly from $[1, 3 * \Delta q_k]$ and is therefore independent from its execution time.

We have generated 20 instances for each different activity set size and we ran our planning algorithms from Section 4.5. In these experiments all activities have the same success rate α and we solved all the instances with rates $[0.2, 0.4, 0.6, 0.8, 1]$. For the best-response algorithm we have used 3 iterations and in each of these iterations the order of agents is randomly determined. Smaller experiments support our choice for 3 iterations: less iterations result in far worse results while more iterations only slightly improve the quality but increase the runtime substantially. Note that we have no guarantee that the best-response approach will converge to an equilibrium at this point. This is an issue that remains open at this point, however early experiments have shown that best-response almost always improves the initial solution at least.

Figure 3 illustrates the total cost obtained for each of the methods under different levels of uncertainty with a box plot. In the plot, the box contains the upper and lower quartile of the result values with the mean shown by the horizontal line. The whiskers show the smallest and largest values and outliers are plotted as red crosses.

The centralised algorithm always computes the social optimal solution in which the total cost is minimal. As to be expected, the individual planning method perform much worse on these instances. Because in this approach the dependencies between agents are ignored, the resulting plan may suffer from high social cost. Indeed this figure shows that the total costs are much higher on average, compared to the central solution.

Using only 3 iterations, the best-response algorithm produces fairly acceptable plans. As we have mentioned before, best-response can been seen as a compromise between individual and central planning. Indeed our experiments show that the total cost is lower on average than when using individual planning, but higher than the centralised method.

5.2 Portfolio Management

For portfolio management we have performed similar experiments. We have generated a set of 5 games for each combination of $|N| \in [2, 5]$, $|E_i| \in [1, 5]$, $|A_i| \in [1, 3]$ and $\beta \in [0.2, 0.4, 0.6, 0.8, 1.0]$ (delay risk is the same for all activities in these instances). We ran our solver on these instances for different values of T. Again we study the worst case in which players are tightly coupled (all activities interfere with at least one of another agent), and we strive to gain insight in the factors contributing to the complexity.

Although exact solving for multiple agents poses a difficult challenge, we have been able to solve several non-trivial instances using the best-response approach. Figure 4 shows the runtime required for several of these instances. These early experiments show that best-responses can be computed in the order of a few minutes for problems where agents are responsible for multiple roads with several activities to choose from.

6. CONCLUSIONS AND CHALLENGES

This paper introduces the practically very relevant problem of infrastructural maintenance planning under uncertainty for selfish agents in a private-values setting. With the help of experts in the field of asset management we developed a model that captures the essence of this coordination problem. Dynamic mechanism design combined with optimally solving MDPs theoretically solves this modelled problem. Through experimental analysis with different encodings in an existing solver, we found that run times become large for more than 5 players, 3 activities, 6 quality levels and 46 time steps (for scenario 1), which is reasonable in practice. For scenario 2, run times for best-response can be computed for multiple agents in a small network. We have thus made an important step towards this practical planning problem, but, additionally we can identify a number of challenges for our community.

In this paper, we used scalar weighting to balance the different objectives in the system. However, asset maintenance planning for infrastructures is inherently a multi-objective problem, even though this has not been acknowledged in procurements until recently. The weighting model has two difficulties. Firstly, it requires accurate and exhaustive operationalisation of objectives in terms of monetary rewards schemes. Secondly, in any practical application, human decision makers are more likely to prefer insight into possible solutions trade-offs over a single black-box solution. In this context, the work by [11] is relevant, in which the authors study approximation techniques for mechanism design on multi-objective problems. Nevertheless, their work has only been applied to static mechanisms. Developing methods combining multi-objective planning under uncertainty with dynamic mechanism design is a hard challenge for the community, but with high potential payoffs in terms of real-world relevance.

Scaling MDP solvers in terms of number of actions has not received large amounts of attention, but is crucial for solving multi-agent problems that suffer from exponential blow up of their action space. Furthermore, the best-response approach that we employed is not guaranteed to converge to the optimal solution, except for special cases such as potential games [14]. Bounding the loss, e.g., by building on those special cases, will provide benefits to the adoption of best-response methods.

Finally, as mentioned in the related work section, approximate solutions often preclude many of the theoretical mechanism-design results to apply. A major challenge here is to identify mechanisms that are more robust to such approximations.

With respect to the implications of our work, it is clear that the planning and coordination of (maintenance) activities in the presence of uncertainty is a complex problem. However, applications exist in several other domains such as bandwidth allocation or smart power grids, and hence the need for a practical solution is high.

The concept of traffic time loss can also be used to stimulate market parties in rethinking current working methods. By adjusting tendering criteria to specific needs on certain areas of the network, bidders can distinguish themselves by offering innovative proposals with limited traffic loss hours. The road authority and several provinces are currently experimenting with this method.

Acknowledgements

This research is part of the Dynamic Contracting in Infrastructures project and is supported by Next Generation Infrastructures and Almende BV. Matthijs Spaan is funded by the FP7 Marie Curie Actions Individual Fellowship #275217 (FP7-PEOPLE-2010-IEF).

7. REFERENCES

[1] S. Athey and I. Segal. An efficient dynamic mechanism. Technical report, UCLA Department of Economics, 2007.

[2] D. Bergemann and J. Valimaki. Efficient dynamic auctions. *Cowles Foundation Discussion Papers*, 2006.

[3] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

[4] C. Boutilier. Planning, learning and coordination in multiagent decision processes. In *Proc. of 6th Conf. on Theoretical Aspects of Rationality and Knowledge*, pages 195–201, 1996.

[5] R. I. Brafman, C. Domshlak, Y. Engel, and M. Tennenholtz. Planning games. In *Proc. Int. Joint Conf. on Artificial Intelligence*, pages 73–78, 2009.

[6] R. Cavallo. Efficiency and redistribution in dynamic mechanism design. In *Proc. of 9th ACM conference on Electronic commerce*, pages 220–229. ACM, 2008.

[7] R. Cavallo, D. C. Parkes, and S. Singh. Optimal coordinated planning amongst self-interested agents with private state. In *Proc. of Conf. on Uncertainty in Artificial Intelligence*, pages 55–62, 2006.

[8] C. d'Aspremont and L. Gérard-Varet. Incentives and incomplete information. *Journal of Public Economics*, 11(1):25–45, 1979.

[9] Der Spiegel. A40: Autobahn nach dreimonatiger sperre freigegeben, 2012. Online, Sep 30.

[10] B. Detienne, S. Dauzère-Pérès, and C. Yugma. Scheduling jobs on parallel machines to minimize a regular step total cost function. *Journal of Scheduling*, pages 1–16, 2009.

[11] F. Grandoni, P. Krysta, S. Leonardi, and C. Ventre. Utilitarian mechanism design for multi-objective optimization. In *Proc. of 21st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 573–584. Society for Industrial and Applied Mathematics, 2010.

[12] T. Groves. Incentives in teams. *Econometrica*, 41(4):617–631, 1973.

[13] J. Hoey, R. St-Aubin, A. Hu, and C. Boutilier. Spudd: Stochastic planning using decision diagrams. In *Proc. of Conf. on Uncertainty in Artificial Intelligence*, pages 279–288, 1999.

[14] A. Jonsson and M. Rovatsos. Scaling up multiagent planning: A best-response approach. In *Int. Conf. on Automated Planning and Scheduling*, pages 114–121, 2011.

[15] J. R. Kok, P. Hoen, B. Bakker, and N. Vlassis. Utile coordination: Learning interdependencies among cooperative agents. In *Proc. Symp. on Computational Intelligence and Games*, pages 29–36, 2005.

[16] F. S. Melo and M. Veloso. Learning of coordination: Exploiting sparse interactions in multiagent systems. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 773–780. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[17] R. Porter, A. Ronen, Y. Shoham, and M. Tennenholtz. Fault tolerant mechanism design. *Artificial Intelligence*, 172(15):1783–1799, 2008.

[18] D. Procaccia and M. Tennenholtz. Approximate mechanism design without money. In *Proc. of ACM Conf. on Electronic Commerce*, pages 177–186, 2009.

[19] M. L. Puterman. *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, 1994.

[20] J. Scharpff, M. T. J. Spaan, L. Volker, and M. M. de Weerdt. Planning under Uncertainty for Coordinating Infrastructural Maintenance. In *Proc. of the International Conference on Automated Planning and Scheduling (ICAPS)*, 2013. To appear.

[21] S. Seuken, R. Cavallo, and D. Parkes. Partially-synchronized DEC-MDPs in dynamic mechanism design. In *Proc. of National Conf. on Artificial Intelligence*, pages 162–169, 2008.

[22] J. Snellen, D. Ettema, and M. Dijst. Activiteitenpatronen en reistijdwaardering (dutch). Technical report, Transumo, December 2007.

[23] R. P. van der Krogt, M. M. de Weerdt, and Y. Zhang. Of mechanism design and multiagent planning. In M. Ghallab, C. D. Spyropoulos, N. Fakotakis, and N. Avouris, editors, *European Conf. on Artificial Intelligence*, pages 423–427, 2008.

[24] L. Volker, J. Scharpff, M. De Weerdt, and P. Herder. Designing a dynamic network based approach for asset management activities. In *Proc. of 28th Annual Conference of Association of Researchers in Construction Management (ARCOM)*, 2012.

[25] D. H. Wolpert, K. Tumer, and J. Frank. Using collective intelligence to route internet traffic. In *Proceedings of the 1998 conference on Advances in neural information processing systems II*, pages 952–958. MIT Press, 1999.