Planning under Uncertainty for Coordinating Infrastructural Maintenance

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

Abstract

We address efficient planning of maintenance activities in 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) 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 maintenance-planning problems and 3) empirical exploration of the complexities that arise in these problems. 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.

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 (commonly referred to as *social cost*) 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.

A powerful real-world example of the benefits of careful maintenance planning 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) (Der Spiegel 2012).

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 quality impact. Furthermore, long-term planning is required to ensure overall network quality. Assuming these uncertainties are known beforehand, as in this work, Markov Decision Processes (MDP) provide a suitable framework to model and solve these types of planning-under-uncertainty problems (Puterman 1994).

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 agents (the 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. Nonetheless, an agent servicing one part of the network also influences agents in other parts as its work has a negative impact on the traffic flow. As a consequence, such congestion based payments may lead to very high throughput penalties for all agents if their maintenance plans are not coordinated 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 contingent, private-values setting. 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 is relatively unexplored territory (Cavallo 2008).

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 (Boutilier 1996) assumes cooperative agents that are willing to disclose private information and share the same utility function. In decentralised MDPs (Bernstein et al. 2002), although execution is decentralised, agents are still assumed to be cooperative and solving Dec-MDPs requires knowledge of all utility functions. 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 (Brafman et al. 2009; Jonsson and Rovatsos 2011; van der Krogt, de Weerdt, and Zhang 2008), 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. The only work we are aware of in this area is by (Detienne, Dauzère-Pérès, and Yugma 2009), 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 cost functions do not have the non-decreasing property.

Another interesting related approach is that of reinforcement learning (Kok et al. 2005; Melo and Veloso 2009) and in particular Collective Intelligence (Wolpert, Tumer, and Frank 1999). In this 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, as these methods cannot provide theoretical guarantees concerning the incentives, they are not adequate in the presence of strategic behaviour, i.e., agents deliberately trying to manipulating the system.

Although stochastic planning has been well studied, only a handful of papers address dynamic mechanism design and/or a combination of the two. Bergemann and Valimaki (2006) proposed a dynamic variant of the VCG mechanism for repeated allocation, implementing the mechanism desiderata in a within-period, ex-post Nash equilibrium. Athey and Segal (2007) studied a dynamic variant of the AGV mechanism (d'Aspremont and Gérard-Varet 1979) that is budget-balanced in the weaker Bayes-Nash equilibrium solution concept. Highly related is the work by Cavallo, Parkes, and Singh (2006), 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. Considering the complexity of the stochastic planning problem we study here, approximation of the planning also seems a viable approach. When resorting to approximate solutions, however, standard theory for strategy-proof mechanisms does not immediately apply (Procaccia and Tennenholtz 2009).

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 several of the 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 approach with uncoordinated agents and best-response playing agents (Section 5). We conclude with a summary of our findings and we 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). 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.

In the infrastructural maintenance planning problem (Volker et al. 2012) 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, etc.).

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 $c_i : A_i \times T \rightarrow \mathbb{R}$ that represents the cost of performing an activity $k \in A_i$ at time $t \in T$. The dependency on time enables modelling of different costs for example for different seasons, or for periods in which 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 its profits are maximised, but plan execution is unlikely to be perfect. Uncertainties in various forms – for example delays, unknown asset states, failures – may be encountered during execution and and hence fixed 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) and states contain all relevant information for its planning problem. We formalise these concepts in Section 3.1, for now it is sufficient to know that we can always observe what activity has been performed by each contractor. We denote the observed activities by P_i and use $P_i(t) = k$ to denote that activity k was performed at time t. Each activity has to be completed before another can be started, therefore there must be exactly d_k time steps for which P_i returns k. Note that an agent can also choose to perform no activity during a time step, which we denote by $P(t) = \circ$ and we assume $\forall t \in T : c_i(\circ, t) = 0$.

Given performed activities P_i, the total revenue W_i agent i will receive is the sum of all w_k for all completed activities k. The total maintenance cost for agent i is given by $C_i(P_i) = \sum_{t \in T} c_i(P_i(t), t)$. 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 revenue 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 performed activities P_i, we can determine the resulting quality state q_e^T at the end of the period T using the recursive formula

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

with (given) initial quality $q_{e_k}^0$. We define the quality payment for agent i after performing activities P_i by $Q_i(P_i) = \sum_{e \in E_i} Q_i(q_e^{|P_i|})$ where $|P_i| = T$ if all performed activities have been observed.

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 P^t denote the set of activities performed by all agents at time t, then the social cost of this combination is captured by $\ell(P^t)$. The impact of an individual agent, given the choices made by others, can be determined by $\ell_i(P^t) = \ell(P^t) - \ell(P^t_{-i})$ in which P^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, possibly based on empirical data.

Recapitulating the above, each agent i is trivially interested in maximising its revenue and minimising its maintenance costs. 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 performed activities P by all players, is given by:

$$u_i(P) = W_i(P_i) - \Big(C_i(P_i) + Q_i(P_i) + \ell_i(P)\Big) \tag{2}$$

in which $\ell_i(P) = \sum_{t \in T} \ell_i(P^t)$. As activity revenues 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.

Recall from the introduction that we are interested in finding socially optimal solutions, but given the individual agent utility of Eq. 2, 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? Moreover, how can we ensure truthful reporting of these private costs? We tackle these questions using dynamic mechanism design.

In the next section we start by discussing how to compute optimal solutions, required to guarantee mechanism truthfulness, 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 (Puterman 1994). 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. The current state of an agent contains all activities that still remain to be performed and its actions are operations to start or continue an activity (explained in detail in Sections 4.3 and 4.4). Important to keep in mind is that the MDP actions $\mathcal{A}_i$ are not equivalent to the agent activities A_i (although in the case of unit-time actions these sets are almost similar).

The function $\tau_i : S_i \times A_i \to \Delta(S_i)$ 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, $r_i : S_i \times \mathcal{A}_i \to \mathbb{R}$ 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. 2). 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 $\pi : S \to \mathcal{A}$ 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 $s \in S$: $\pi^*(s) = \arg\max_{\pi \in \Pi} V^0(\pi, s)$ with

$$V^{t_0}(\pi, s) = \mathbb{E}\Big[\sum_{t=t_0}^{\infty} \gamma^t r(s^t, \pi(s^t))) \mid s^{t_0} = s\Big] \quad (3)$$

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 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 optimal planning under uncertainty, they assume global knowledge of all 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 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 (Groves 1973) 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 to take (in expectation) 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 (Bergemann and Valimaki 2006; Cavallo 2008). For dynamic-VCG the decision policy is required to be optimal, i.e., the one maximising the reward of all players, 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 it incurs to 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. 3) 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

$$\sum_{j \neq i} r_j(s^t, \pi^*(s^t)) + V^{t+1}(\pi^*_{-i}, s^{t+1}) - V^t(\pi^*_{-i}, s^t) \quad (4)$$

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 optimal policies and finally we determine the payments of the mechanism, as discussed in the previous section.

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 (Hoey et al. 1999) 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

Finding an efficient joint policy π^* that maximises the sum of all agent utilities u_i (Eq. 2) 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 then 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 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 to 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. Crucial is that these variables can be set independently 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.

	repeat	duration d_k	success prob. α_k	delay duration h_k	delay prob. 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. These parameters are explained in Section 4.3 and 4.4.

The action chain encoding exploits the fact that we can decide on an action for each player sequentially, instead of having to decide on them all at once (as with enumeration). Through the use of a player token, each agent gets a 'turn' to determine its 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 sets 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. Through 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 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. 2) 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 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.

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 its 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 later; 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 by 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 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 (Jonsson and Rovatsos 2011).

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 downsides of this approach are that we will have to settle for Nash equilibria (if they exist) and the ordering of agents matters.

5 Evaluation

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 with a time limit of 3 hours per instance, except for the experiments of Section 5.2 which had a time limit of one day.[1]

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 difficulty 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. For the social costs ℓ we

[1]The testset is available at http://www.alg.ewi.tudelft.nl/fileadmin/alg/homepages/scharpff/icaps-testset.rar

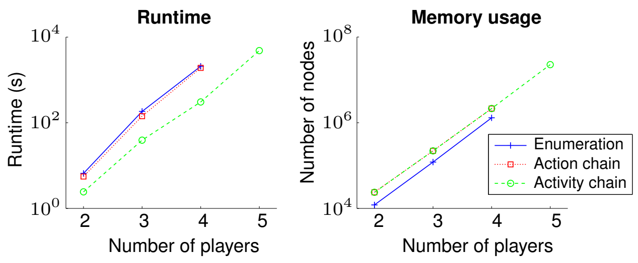

Figure 1: Comparison of runtime (left) and memory use (right) for different encoding methods and player set sizes, $|A_i| = 3$, $|T| = 46$, $|Q| = 6$ (both log scale).

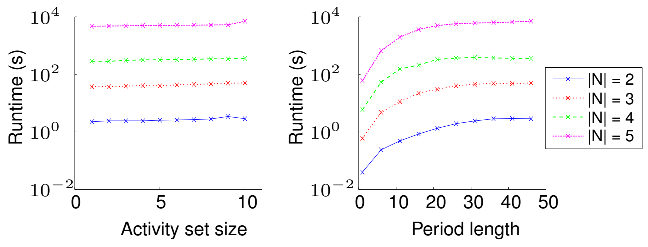

Figure 2: Runtimes for different activity set sizes $|A_i|$ with plan period length $|T| = 46$ (left), and different $|T|$ with $|A_i| = 10$ (right) using activity chains (both log scale).

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. 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 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.

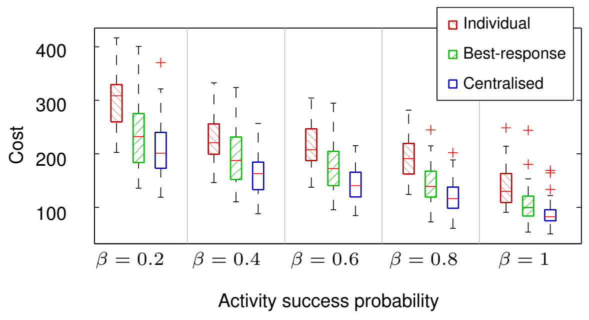

Figure 3: Total cost using different planning approaches for the activities with failure problems (lower is better).

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. In each instance, the activities share the same success rate $\alpha = [0.2, 0.4, 0.6, 0.8, 1]$ for all activities. For the best-response algorithm we have used 3 iterations with random agent orderings. 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, however early experiments have shown that best-response almost always improves the initial solution.

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 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

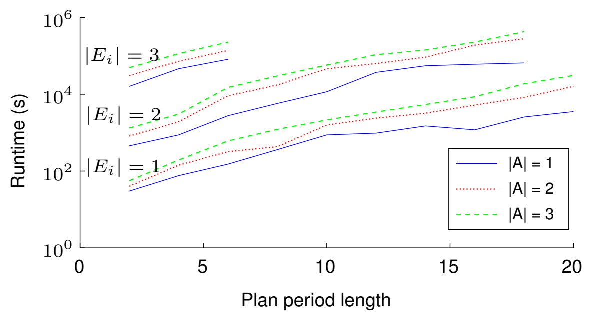

Figure 4: Runtimes of best-response planning for portfolio management for various road set sizes $|E_i|$, activities per road $|A|$ and plan length $|T|$ (log scale). The cut-off for $|E_i| = 3$ at $|T| = 6$ is due to the time limit of 1 day.

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 at this point, we have been able to develop joint plans for several non-trivial instances using the best-response approach. Figure 4 illustrates the runtime required for finding an optimal response, given the planning choices made by others, for various road set, activity and plan period sizes. 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, but also that it quickly becomes intractable for larger plan horizons and road set sizes.

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 maintenance planning 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 but might be difficult in practical scenarios. Through experimental analysis with different encodings in an existing solver, we found that we can solve practical examples of scenario 1 within reasonable time. 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, and identified 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 Grandoni et al. (2010) 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 received relatively little 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 (Jonsson and Rovatsos 2011). 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 Dutch road authority and several provinces of The Netherlands are currently experimenting with this method in the Netherlands.

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).

References

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

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

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

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

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

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

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

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

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

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

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

Groves, T. 1973. Incentives in teams. *Econometrica: Journal of the Econometric Society* 617–631.

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

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

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

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

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

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

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

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

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