Computers & Industrial Engineering 126 (2018) 578–594

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Reward-based Monte Carlo-Bayesian reinforcement learning for cyber preventive maintenance

Theodore T. Allen[a,*], Sayak Roychowdhury[b], Enhao Liu[a]

[a] The Ohio State University, Integrated Systems Engineering, 1971 Neil Avenue – 210 Baker Systems, Columbus, OH 43221, United States
[b] Indian Institute of Technology, Kharagpur, Industrial and Systems Engineering, Kharagpur 721302, India

ARTICLE INFO

Keywords:
Preventative maintenance
Cyber security
Markov decision processes
Parametric uncertainty

ABSTRACT

This article considers a preventive maintenance problem related to cyber security in universities. A Bayesian Reinforcement Learning (BRL) problem is formulated using limited data from scan results and intrusion detection system warnings. The median estimated learning time (MELT) measure is introduced to evaluate the speed at which a control system effectively eliminates parametric uncertainty and probability is concentrated on a single scenario. It is demonstrated that the Monte Carlo BRL with enhancements including Latin hypercube sampling (LHS) to generate scenarios, identical systems multi-task learning, and reward-based learning achieves shorter MELT values, i.e., "faster" learning, and improved objective values compared with alternatives in a numerical study. Rigorous results establish the optimality of the derived control strategies and the fact that optimal learning is possible under steady state assumptions. Also, the real-world case study of policies for patching Linux critical server cyber vulnerabilities generates insights including the potential to reduce expenditure per host by mandating compensating controls for critical vulnerabilities.

1. Introduction

Cyber security is a major source of losses for organizations of all types and is regarded as the top domestic and international security risk (Clarke and Papadopoulos, 2016). The costs for cyber security losses to organizations are expected to exceed $0.5T in 2017 with costs per organization averaging over $11M (Ponemon Institute, 2017). Cyber maintenance relates to the accumulation of so-called "vulnerabilities" which are discovered in existing software and found in periodic scans. Once vulnerabilities are discovered, patches to eliminate them may take time in transactions, research, testing, and/or implementation. Also, new exploits may become available before the elimination of the vulnerability to permit intrusions.

More than ninety percent of known intrusions or warnings from intrusion detection systems relate to hosts with known vulnerabilities (Cockburn, 2009, Afful-Dadzie and Allen, 2016). Vulnerabilities have different levels of severity as assigned by the common vulnerability scoring systems (CVSS) and a common policy is to enforce patching or remediation of high and critical vulnerabilities while ignoring low and medium vulnerabilities with little regard to operating system or system role (Afful-Dadzie and Allen, 2014; Allen, Sui, and Parker, 2017). Afful-Dazie and Allen (2014) formulated the maintenance problem as a Markov decision process while noting the importance of addressing parametric uncertainty because of the limited amount of data (if any) about policies relating to patching low and medium vulnerabilities. Here, we seek to develop methods that can mitigate the effects of parametric uncertainty and reduce the expected discounted maintenance costs.

Bayesian Reinforcement Learning (BRL) methods generalize the Markov decision process to address limited data, i.e., parametric uncertainty (see Ghavamzadeh, Mannor, Pineau, and Tamar, 2015, for a recent review). Reinforcement Learning algorithms can be broadly categorized in *model-based* and *model-free* methods (Sutton and Barto, 1998; Ghavamzadeh, et al., 2015). Model-based algorithms first fit a model offline over historic data, states and actions are manually identified, and then decisions are made online. In model-free methods, the agent directly learns about the states and actions through interactions with the environment and taking actions simultaneously. Bayesian Reinforcement Learning applies the techniques of Bayesian inference using prior knowledge of the problem domain. Parametric uncertainty is modeled using appropriate probability distributions, and decisions are made through maximizing expected reward. A number of model-based methods leverage the structure of Markov Decision Process (MDP), for instance, Duff (2002) introduced Bayes-Adaptive Markov Decision Process (BAMDP). Here, we seek to establish formally the link between finite scenario BRL and Partially Observable Markov Decision

* Corresponding author.
E-mail addresses: allen.515@osu.edu (T.T. Allen), sroychowdhury@iem.iitkgp.ac.in (S. Roychowdhury), liu.5045@osu.edu (E. Liu).

https://doi.org/10.1016/j.cie.2018.09.051
Received 20 February 2018; Received in revised form 26 September 2018; Accepted 27 September 2018
Available online 29 September 2018
0360-8352/

Processes (POMDP), where the uncertain model parameters are sampled from a prior distribution and projected to a discrete hybrid state space. Poupart, Vlassis, Hoey, and Regan (2006) introduced BEETLE algorithm, which extends the POMDP solver PERSEUS (Spaan and Vlassis, 2005) to solve the BRL problem. Generally, model free methods are relevant when data access is inexpensive and model-based are relevant in cases such as ours when each data point is associated with significant potential costs.

Wang, Won, Hsu, and Lee (2012) proposed Monte Carlo-Bayesian Reinforcement Learning (MC-BRL), which uses finite numbers of scenarios to address parametric uncertainty, a choice motivated by simplicity and computational efficiency. Wang, Won, Hsu, and Lee (2012) note that the many MC-BRL methods are limited because of the need for Bayesian conjugacy which is restricted to learning single transition-at-a-time and not from rewards. Here, we extend MC-BRL to address cyber vulnerability maintenance to include:

1. *Variation Reduction Methods and Few Variables for Scenario Selection*
2. Common BRL formulations assume that *all* transition probabilities are uncertain, and scenarios are selected arbitrarily or randomly (Ghavamzadeh, et al., 2015). However, uncertainty in maintenance concentrates on the specific combinations of unobserved states and actions. Variation reduction techniques from Monte Carlo simulation (e.g., Tang, 1993, Allen, 2011, Chapter 8) can reduce the number of scenarios needed to approximate infinity.
3. *Exact Multi-Task Learning for Identical Systems*
4. A recent survey by Zhang, Yang (2017) explores different types of "Multi-task Learning" (MTL). The MTL methods try to leverage information from multiple related task to enhance the performance of all the tasks. According to this survey, MTL methods can be classified in different categories including feature based learning (Argyriou, Evgeniou and Pontil, 2008), low-rank approach (Ando and Zhang, 2005, Maurer, Pontil, and Romera-Paredes, 2013), task clustering (Bakker and Heskes, 2003), task relation learning (Evgeniou, Micchelli and Pontil, 2005, Kato, Kashima, Sugiyama, and Asai, 2010) dirty approach (Jalali, Ravikumar, Sanghavi, and Ruan, 2010), multi-level approach (Jawanpuria and Nath, 2012) and deep learning (Li, Liu and Chan, 2015, Misra, Shrivastava, Gupta and Hebert, 2016). Here, we explore identical systems MTL reinforcement learning which is new.

Only a subset of "multi-task" BRL applications have focused on learning many transitions at-a-time (Ghavamzadeh, et al., 2015). Wilson, Fern, Ray, and Tadepalli (2007) used multi-task reinforcement learning (MTRL) to solve a sequence of MDPs chosen randomly from a fixed but unknown distribution. They modeled the MDPs with hierarchical Bayesian Mixture Model to account for parametric uncertainty. Taylor and Stone (2009) surveys transfer learning algorithms and discusses their possible application in the domain of RL. Calandriello, Lazaric and Restelli (2014) explores the case where all tasks can be represented in the linear approximation space of a subset of features and proposes two algorithms, which are multi-task extensions of the fitted Q-iteration algorithm. Parisotto, Ba and Salakhutdinov (2016) proposes the "Actor-Mimic" method which uses deep reinforcement learning and model compression techniques for training a single network, which can then be implemented for different tasks leveraging the help of "experts".

These representations of deep policy networks are shown to be generalized for new tasks without prior expert knowledge. Our work here may be viewed as a simple version of multi-task BRL methods in which subsystems are identical (Lazaric and Ghavamzadeh, 2010). Our contribution relates to proving that the joint policy is optimal for these multitask networks and their application to production systems. Specifically, maintenance problems involve sub-networks of effectively identical hosts which can be computers, servers, printers, or other devices. By controlling multiple systems simultaneously, "fast" learning is enabled which we quantity in terms of the number of periods until parametric uncertainty is largely eliminated.

5. *Expected Reward Uncertainty:* The expected costs of selected actions in specific states are uncertain because of limited data since those actions have not been applied previously. Since rewards are continuous numbers, observing rewards generally provides an opportunity to reduce parametric uncertainty in shorter time periods than single transition observations. Thus, observations of rewards result in "fast" learning. Apparently, this is new for model-based BRL (Ray and Tadepalli, 2010).

Previous BRL research is based on the realization that MC-BRL is a partially observable Markov decision processes (POMDPs) observed without proof in Duff (2002). One of the objectives here is to provide a rigorous proof of this equivalence. Poupart, Vlassis, Hoey, and Regan (2006) provided rigorous results for the infinite scenario or continuous uncertainty case involving only transition-based learning, i.e., they did not employ reward-based learning as we explore here. Also, we seek to quantify the rate of learning and illustrate the relative benefits of reward-based and identical systems-based learning.

Applications of POMDPs to production system problems are of increasing interest (Srinivasan and Parlikad, 2013; Jafari and Makis, 2016; Alagoz, Ayvaci, and Linderoth, 2015; Cheng, Zhou, and Li, 2017). In particular, Srinivasan and Parlikad (2013) explored POMDP for maintenance decision-making and monitoring. Yet, using POMDP or BRL to model and reduce parametric uncertainty in production systems problems is unexplored. An objective of this research is to illustrate the benefits of such methods in production systems decision-making.

Our case study features subnetworks of Linux hosts in different roles within a large midwestern university. The university scans entities called "hosts" with fixed IP addresses monthly and requires patching or remediation for high and critical-level vulnerabilities within thirty days. Prior to our project, the university applied (approximately) the same approach to all hosts. Also, the university granted administrative privileges to many faculty and staff without charge. We use 21 months of scan and associated intrusion system data. Through interviews, we obtained cost estimates for patching, remediation, incident investigations, and actual intrusions. Our applied objective included prioritizing which additional vulnerabilities to inspect and estimating the value of administrative privileges.

This paper is organized as follows. In Section 2, we present the assumptions and the key elements of the problem along with a mathematical programming formulation. In Section 3, solution methods are described including the conversion of the BRL problem into a POMDP formulation and point-based POMDP solution methods are reviewed. In Section 4, rigorous results are also provided for MC-BRL establishing the optimality of solutions and the relevance of steady state conditions. In Section 5, a computational experiment is described investigating the effects of learning enablers on values and learning rates. Section 6 describes the case study problem and findings for Linux critical hosts. Section 7 discusses the practical results, limitations, and opportunities for future research.

2. Problem definition

Bayesian reinforcement learning (BRL) problems in the literature are often expressed in terms of a single system subject to the control policy with an infinite number of scenarios. In our problem, we have multiple systems under control and a focus on finite numbers of scenarios. As noted by Wang, Won, Hsu, and Lee (2012), using finite numbers of scenarios simplifies BRL and generally reduces training computation times. Finite number of scenarios also have implications for reward-based and identical system-based "fast" learning that we explore in Sections 3–5.

In cyber maintenance, often a central authority sets the policy and administrators from various units use fractions of their work times to search for vulnerabilities and, if found, test and apply patches. Testing is important in some cases because the patches can disrupt functionality, for instance specific hardware and software, may lose functionality and compliance. If no patch is applied, restricting access to the internet or other actions can remediate the vulnerabilities. At the same time, failing to remediate the vulnerabilities can result in intrusions occurring or being worsened as attackers gain greater access. Even if the hosts have no important information or access to be lost, they can become sources for further attacks and incident investigations can be expensive.

Similar tradeoffs between repair actions, e.g. patching vulnerabilities, and breakdown costs, are relevant to other systems. For example, manufacturing machines may degrade over time potentially requiring repair and failures could cause injuries or other losses. In medical domains, patient ailments could worsen or even become fatal if no attention is given.

2.1. The objective function

The primary objective is to minimize the expected discounted losses in investigation, remediation and maintenance cost, over a long or infinite future, summed over a set of sub-systems which could be hosts (computers, servers, printers, …). Remediation actions affect the trajectory of the system. Learning about transition probabilities and expected rewards makes it impossible to assume that an optimal policy would have the same action for every equivalent scan result. Parametric uncertainty is addressed using expectation over the model scenarios (Wang, Won, Hsu, and Lee, 2012).

Note that previous researchers have written the Bayesian Reinforcement Learning formulation recursively, which effectively assumes that the system is in steady state for all relevant cases. We seek to write a general optimization formulation that does not assume the system is always in steady state and accounts for multiple identical systems. For additional details about cyber maintenance costs, see, Afful-Dadzie and Allen (2016), and Hou (2015).

2.2. The mathematical model

We use the following notation for the problem parameters and variables.

Parameters:

N	Total number of "natural" system states (e.g., states of degradation)
u	Number of feasible actions
A	Set of actions with values $a = 1, \ldots, u$
H	Number of periods or epochs (often $H = \infty$)
W	Number of subsystems (differing only possibly by the current state)
$n_{i,t}$	Number of subsystems in state i in period t
γ	Discount factor (e.g., 0.99 for monthly time periods)
q	Number of model scenarios
z	Number of observation levels
g	Number of possibly observed reward levels
m	Number of independent uncertain parameters
$p^a_{ij(k)}$	Probability of a transition from states i to j under action a & scenario k ($\boldsymbol{p}^a_k$ matrix form)
$r^a_{ij(k)}$	Expected reward of a transition from states i to j under action a & scenario k ($\boldsymbol{r}^a_k$ matrix form)
$r^H_{j(k)}$	Expected reward in the last period (H) for being in state j under scenario k
o^a_{iO}	Probability of observing O when the state is i under action a
Y_{0w}	Initial state of subsystem w
σ	Standard deviation of the rewards
L_g	Lower limit of loss interval indexed by g
D_{kj}	Randomize Orthogonal Array Latin Hypercube for scenario k and parameter j
α^a_{ij}	Historical count of transitions from state to i state j under action a

Variables:

$Y_{tw}(k, a)$	State of subsystem w in period t (distribution depends on scenario k and action a)
$p_t(k, O_{11}\ldots O_{tW})$	Probability that the true system is modeled by scenario k in period t
b_{ti}	Probability of being in state i in period t (belief state, $\boldsymbol{b}_t$ in vector form)
O_{tw}	Observation in period t from system w
ϵ_{tw}	Random deviation of the reward from the mean in period t for system w (with $E[\epsilon_{tw}] = 0$)
G	Observed discrete reward level (truncated to g possible levels values of $r^a_{ijk} + \epsilon$)
$V_t(i, k)$	Expected summed discounted value in state i in period t under scenario k
$V_t(\boldsymbol{b}_t, k)$	Expected summed discounted value in belief state $\boldsymbol{b}_t$ in period t under scenario k
$V_1(\boldsymbol{b}_t, p_1(1), \ldots p_1(q))$	Expected summed discounted value in belief state $\boldsymbol{b}_t$ in period t
$\pi(O_{11}\ldots O_{tW})$	Mapping $O_{11}\ldots O_{tW} \rightarrow A$ representing the system control policy for each subsystem
$\widetilde{U}_{kj}$	Orthogonal Array Latin Hypercube samples for scenario k and parameter j

To our knowledge, the model-based Bayesian Reinforcement Learning formulation has not been documented completely in prior works in a way that accounts for the values of future observations (e.g., see Sutton and Barto, 1998, Wiering and Van Otterlo, 2012). Based on the above assumptions, we developed the following BRL model to derive the optimal control policy, $\pi(O_{11}\ldots O_{tW})$:

$$\textit{Minimize } V_1(\boldsymbol{b}_1, p_1(1), \ldots p_1(q), H) = \sum_{w=1}^{W}\sum_{k=1}^{q} p_1(k, O_{11}\ldots O_{1W}) \underset{Y_{11(k,a)},\ldots Y_{H,W}(k,a)_{11},\ldots}{\mathrm{E}} \left[\gamma^H\left(r^H_{Y_{Hw}(k,a)} + \epsilon\right) + \sum_{t=1}^{H-1}\gamma^{t-1}\left(r^{\pi(O_{11}\ldots O_{tW})}_{Y_{(t-1)w}(k,a)Y_{tw}(k,a)} + \epsilon_{tw}\right)\right] \quad (1)$$

$$\text{Subject to: } Y_{tw}(k, a)|Y_{(t-1)w}(k) \sim \textit{Multinomial}\left[1, \left(p^a_{Y_{(t-1)w}(k)1k}, \ldots, p^a_{Y_{(t-1)wN(k)Nk}}\right)\right], \quad (2)$$

$$Pr(O_{tw}|a, w, i) = o^a_{iO} \quad \forall\, a = 1, \ldots, A;\ w = 1, \ldots, W;\ i = 1, \ldots, N,\ O_{tw} \in Y_{tw}(k, a), G, \quad (3)$$

$$\text{and } Y_{0w}(k, a) = Y_{0w} \quad \forall\, w = 1, \ldots, W;\ t = 1, \ldots, H;\ k = 1, \ldots, q;\ a = 1, \ldots, u.$$

Therefore, Eq. (1) gives the expected discounted sum of rewards by including a weighted sum over the model scenarios reflecting parametric uncertainty. Eq. (2) represents single multinomial samples generating the underlying Markov chains. Note that here we emphasize the infinite horizon $H = \infty$ case, but results hold for the finite horizon cases also. Eq. (3) is only needed for partially observable systems, i.e.,

Table 1
(a) Observed transition counts, (b) full factorial, (c) pseudorandom uniform deviates, (d) OALH deviates, (e) gamma deviates for Dirichlet deviates, (f) derived scenarios from scaling the gamma deviates.

(a)

from\to	1	2
1	8	2
2	6,000	4,000

(b)

1	1
2	1
1	2
2	2

(c)

0.08	0.75
0.45	0.91
0.82	0.27
0.89	0.42

(d)

0.04	0.38
0.73	0.46
0.41	0.64
0.95	0.71

(e)

3.80	1.31
9.44	1.54
7.06	2.16
12.97	2.49

(f)

0.74	0.26
0.60	0.40
0.86	0.14
0.60	0.40
0.77	0.23
0.60	0.40
0.84	0.16
0.60	0.40

the $Y_{tw}(k, a)$ are not observable. The concept of partial observability is important because we demonstrate that model scenarios induce a type of partial observability.

The objective in Eq. (1) balances repair or patching and incident response costs. Repairs influence the trajectory of the hosts through the Markov chains in constraint set (2). The expected rewards r^{π}_{ijw} reflect repair and response costs as dictated by the policy, π. While the general notation may seem unfamiliar, it is important to note the familiar special cases. With $W = 1$ subsystem, $q = 1$ scenario, and $O_{t1} = Y_{tw}(k, a)$, the single system has no parametric uncertainty and is fully observable. Then, the formulation reduces to ordinary Markov decision processes (Puterman, 2014). For MDP, it is well known that the current state is a sufficient statistic. Therefore, the optimal policies can be derived using a recursive approach and their dependence can be written $\pi(O_{11}...O_{t1}) = \pi(Y_{t1}(1, 1))$.

With $W = 1$ subsystem, $q = \infty$ scenarios, $H = \infty$ time periods, and $O_{t1} = Y_{tw}(k, a)$, there is a single subsystem with infinity scenarios (continuous parametric uncertainty), and an infinite horizon. This is the focus of the Bayesian Reinforcement Learning (BRL) literature (Ghavamzadeh, et al., 2015). Also, Monte Carlo-Bayesian Reinforcement Learning (MC-BRL) relates to cases with $W = 1$ and $q < \infty$. Here, we introduce the $W > 1$ case and consider more general observations than previous BRL investigations. We also focus on the MC-BRL$q < \infty$ cases in part for computational tractability. Specifically, we focus on observations which are a combination of the subsystem state and the discrete reward level, i.e., $O_{tw} = \{Y_{t+1w}(k, a), R\}$. Note that Ross, Pineau, Chaib-draa, and Kreitmann (2011) consider general observations, yet they do not consider $W > 1$ subsystems, reward-based learning specifically, and computationally efficient possibilities through using specific types of finite scenario formulations as we seek to do.

The case with $W = 1$ subsystem, $q = 1$ scenario, and $O_{t1} \neq Y_{tw}(k, a)$ has relevance to the solution methods here. Then, the formulation reduces to Partially Observable Markov Decision Problems (POMDP, e.g., see Smallwood and Sondik, 1973, Spaan and Vlassis, 2005). For POMDP, Smallwood and Sondik (1973) showed that the belief state, $\boldsymbol{b}_t$, is a sufficient statistic. Therefore, the optimal policy dependence can be written $\pi(O_{11}...O_{t1}) = \pi(\boldsymbol{b}_t)$. The solution methods that we consider here involve converting the formulation in Eq. (1) which is an identical systems BRL with reward-based learning to a specific POMDP formulation. Then, the POMDP methods can create exact or approximate policy solutions depending on the problems size and speed requirements. The rigorous results relate to proving that solving optimally a specific POMDP formulation will solve optimally Eq. (1).

2.3. Scenario generation

The formulation in Eqs. (1), (2), and (3) is expressed in terms of known fixed model scenarios indexed by k. In practice, the scenarios must be generated using the available data relating to transition probabilities and expected rewards. Fortunately, sufficient data is typically available about specific state-action combinations such that parametric uncertainty in the associated transition probabilities and expected rewards can be ignored. For example, in cyber maintenance an organization might have millions of data points related to patching or repair attempts for hosts with high and critical vulnerabilities. At the same time, it might have little or no data about patching for hosts with only low and medium-level vulnerabilities. Here, we focus on the assumption of, at most, m independent parameters. While the q scenarios in Equation (1) may differ by more than m parameters, the differences derive from only m values.

Using $q < \infty$ scenarios has benefits including the possibility of reward-based learning described in the next section. Yet, the solution of the infinite scenario problem could be of interest. Also, often there is flexibility for generating scenarios including the parameter q. Tang (1993) proposed an unbiased estimate of an expected value (potentially infinite sum) using Orthogonal Array Latin Hypercube (OALH) sampling. OALH methods generalize Latin hypercube sampling from McKay et al. (1979). We generate q scenarios using independently distributed deviates, $\widetilde{U}_{kj}$:

$$\widetilde{U}_{kj} = \frac{D_{kj} - 1 + U_{kj}}{q} \text{ where } U_{kj} \sim IIDU[0, 1] \quad \forall\, k = 1, ..., q;\ j = 1, ..., m. \tag{4}$$

where D_{kj} is the input from run k and parameter j from an Latin hypercube (in our examples) or OALH similar to that used in Roychowdhury (2017).

Consider a two-state example with $m = 2$ parameters: p^1_{11k} and p^1_{12k}. This is described in Table 1 for action 1. There is a significant amount of data about transitions starting in state 2 so the parametric uncertainty may be ignored. There is much less data starting in state 1 (Table 1a). Using a full factorial (Table 1b) and random uniform deviates (Table 1c), deviates are generated using Eq. (4) in Table 1d.

Then, the deviates are converted using the inverse gamma distribution (Table 1e) and four scenarios are generated using scaling to generate Dirichlet deviates (Table 1f) which provide $p^1_{11(1)}, ..., p^1_{22(4)}$. Tang (1993) helped to establish that this approach reduces the variance of the estimates generated in relation to a $q = \infty$ version of Eq. (1). In our examples, we use the deviates, $\widetilde{U}_{kj}$, as inputs to generate Dirichlet sampled random probabilities and normally distributed expected mean rewards. Following Delage and Mannor (2010), we add a constant to the counts to smooth the estimates. Having fewer scenarios focusing on fewer (relatively) uncertain parameters promises to accelerate learning.

3. Solution methods

In this section, we propose methods to solve the formulation in Eqs. (1), (2), and (3). The overall approach is to convert multiple "identical

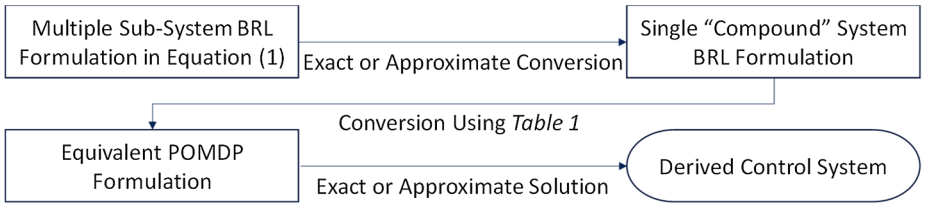

Fig. 1. Solution method involves two conversions and then the POMDP solver.

system" problems to create a single "compound" system. The compound system is another MC-BRL problem with different parameters indexed in our notations by a single "~" symbol.

The compound system is then transformed into an equivalent POMDP problem and solved. The parameters of the POMDP formulation are associated with two "~" symbols. In our examples, we solve the POMDP approximately using the PERSEUS point-based solution method. A key aspect is the development of so-called "observation matrices" used in Bayesian updating. The outline of the solution methods is shown in Fig. 1. Overall, we follow previous MC-BRL research such that the effective or "pure" state, i, for a subsystem is a combination of the ordinary state, Y, and the scenario k (Wang, Won, Hsu, and Lee, 2012, Hou, 2015).

3.1. Identical subsystems and the "Compound" system

Much of the Bayesian Reinforcement Learning (BRL) literature uses autonomous robot examples to illustrate the methods. Here, we focus on policy development which could affect large numbers of subsystems. For example, a cyber-maintenance policy could control a computer cluster of ten effectively identical computer hosts imaged together but in different states. Also, manufacturing lines could have approximately identical robots offering an opportunity to accelerate learning about the merits of repair actions.

The approach explored here creates "compound" systems which relate to the combination of subsystems. For example, if we have $W = 2$ systems each with $q = 2$ states, then there are $N_c = 4$ compound states $(Y_{t1} = 1, Y_{t2} = 1)$, $(Y_{t1} = 2, Y_{t2} = 1)$, $(Y_{t1} = 1, Y_{t2} = 2)$, and $(Y_{t1} = 2, Y_{t2} = 2)$, The multiple subsystems formulation in Eqs. (1), (2), and (3) with$W > 1$ can become a single subsystem $W = 1$ formulation with the following notation for the compound system.

Parameters:

$\widetilde{N}$	Total number of "compound" system states.
$\widetilde{p}^{a}_{\widetilde{i}\widetilde{j}(k)}$	Probability of a transition from compound states $\widetilde{i}$ to $\widetilde{j}$ under action a and scenario k ($\widetilde{\boldsymbol{p}}^{a}_{k}$)
$\widetilde{r}^{a}_{\widetilde{i}\widetilde{j}(k)}$	Expected reward of a transition from compound states $\widetilde{i}$ to $\widetilde{j}$ under action a and scenario k ($\widetilde{\boldsymbol{r}}^{a}_{k}$)
$Y(w)\|\widetilde{i}$	The state index for subsystem w based on the definition of compound state $\widetilde{i}$

Variables:

$\widetilde{Y}_t$	Compound state of subsystem period t
$n_{it}(\widetilde{i})$	The number of subsystems in state i in period t for a system in compound state $\widetilde{i}$
$\pi(O_{11}...O_{tW})$	Mapping $O_{11}...O_{tW} \rightarrow A$ with arbitrary selections if actions differ for subsystems at the same natural state.

The expected rewards and transition probabilities can be derived by a simple calculation based on the individual subsystem properties. The compound system transition matrices are

$$\widetilde{p}^{a}_{\widetilde{i}\widetilde{j}(k)} = \prod_{w=1}^{W} p^{a}_{Y(w)|\widetilde{i}Y(w)|\widetilde{j}(k)} \quad \forall\, \widetilde{i}, \widetilde{j} = 1, ..., \widetilde{N};\ k = 1, ..., q;\ a = 1, ..., u. \tag{5}$$

For example, with $N = 2$ and $W = 2$ possible identifications, the compound transition matrix for action a is given in Table 2. The ordering was given to illustrate the intuitive construction of the compound states.

Similarly, the compound expected reward is a simple sum over the subsystem expected rewards:

i.e., (system 1 is in state 1, system 2 is in state 2) and (system is in state 2, system 2 is in state 1). Then, the expected rewards can be derived by a sum over the possible identifications or assignments

$$\widetilde{r}^{a}_{\widetilde{i}\widetilde{j}\,k} = \sum_{w=1}^{W} r^{a}_{Y(w)|\widetilde{i}Y(w)|\widetilde{j}(k)} \quad \forall\, \widetilde{i}, \widetilde{j} = 1, ..., \widetilde{N};\ k = 1, ..., q;\ a = 1, ..., u. \tag{6}$$

In some situations, it may be helpful to collapse the states using the identical property of the subsystems. For example, the compound state $(Y_{t1} = 2, Y_{t2} = 1)$ may be regarded as effectively equivalent to $(Y_{t1} = 1, Y_{t2} = 2)$. For these situations and specializing to the $N = 2$ special case, the two state success counts are the sums of binomial random variables which are Poisson binomial distributed (Chen and Liu, 1997). Yet, in practice, actions need to be assigned to each specific subsystem. Therefore, it can be expedient to model the subsystem states maintaining their identifiability.

3.2. Transformation to a POMDP

Duff (2002) proposed that the formulation in Eqs. (1), (2), and (3) with $q = \infty$ scenarios could be solved (approximately) by converting the problem to a Partially Observable Markov Decision Problems (POMDP). In the converted POMDP, the "pure" states, as we have designated them, are combinations, in our terminology, of the ordinary

Table 2
Compound transition probability matrix with $\boldsymbol{N} = 2$ states and $\boldsymbol{W} = 2$ subsystems.

From State\To State	$(Y_{t1} = 1, Y_{t2} = 1)$	$(Y_{t1} = 2, Y_{t2} = 1)$	$(Y_{t1} = 1, Y_{t2} = 2)$	$(Y_{t1} = 2, Y_{t2} = 2)$
$(Y_{t1} = 1, Y_{t2} = 1)$	$\left(p^{a}_{11(k)}\right)^2$	$p^{a}_{12(k)}p^{a}_{11(k)}$	$p^{a}_{11(k)}p^{a}_{12(k)}$	$\left(p^{a}_{12(k)}\right)^2$
$(Y_{t1} = 2, Y_{t2} = 1)$	$p^{a}_{21(k)}p^{a}_{11(k)}$	$p^{a}_{22(k)}p^{a}_{11(k)}$	$p^{a}_{21(k)}p^{a}_{12(k)}$	$p^{a}_{22(k)}p^{a}_{12(k)}$
$(Y_{t1} = 1, Y_{t2} = 2)$	$p^{a}_{11(k)}p^{a}_{21(k)}$	$p^{a}_{12(k)}p^{a}_{21(k)}$	$p^{a}_{11(k)}p^{a}_{22(k)}$	$p^{a}_{12(k)}p^{a}_{22(k)}$
$(Y_{t1} = 2, Y_{t2} = 2)$	$\left(p^{a}_{21(k)}\right)^2$	$p^{a}_{22(k)}p^{a}_{21(k)}$	$p^{a}_{21(k)}p^{a}_{22(k)}$	$\left(p^{a}_{22(k)}\right)^2$

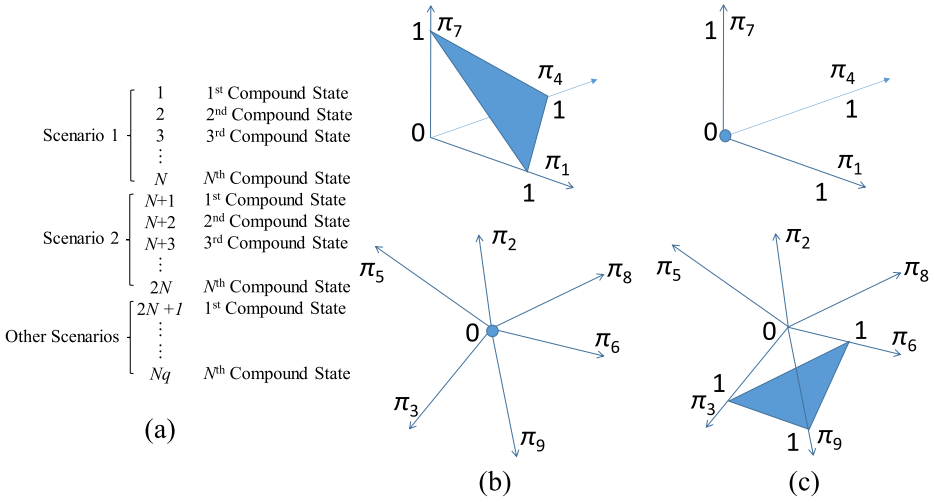

Fig. 2. (a) States of the equivalent POMDP and feasible region after (b) and (c).

"natural" state, $\widetilde{Y}_t$, and the model scenario (see Fig. 2a). The pure state is not observable since the decision-maker has uncertainty about the model scenario. Also, the nonzero probabilities only exist in quadrants of the simple as indicated in Fig. 2(b) and Fig. 2(c). Further, the values of the nonzero probabilities correspond to the current scenario probabilities, $p_t(k, O_{11}...O_{tW})$.

The BRL and POMDP correspondence has motivated much research (see Ghavamzadeh, et al., 2015). Yet, for the $q = \infty$ case, the correspondence is to a POMDP with infinity states. Hou (2015) documented our joint proof of the correspondence for arbitrary q. In this section, we extend the proof to compound systems and include additional details. Therefore, the result here more completely proves the exact correspondence alluded to in general in Duff (2002) and in Wang, Won, Hsu, and Lee (2012).

Parameters:

$\boldsymbol{B}$	Set of probability vectors or "reachable" belief states often generated from simulations
$\widetilde{\widetilde{N}}$	Total number of "equivalent POMDP" system states
$\widetilde{\widetilde{p}}^{a}_{\tilde{i}\tilde{j}k}$	Probability of a transition from states $\tilde{i}$ to $\tilde{j}$ under action a and scenario k ($\widetilde{\widetilde{\boldsymbol{p}}}^{a}_{k}$ matrix form)
$\widetilde{\widetilde{r}}^{a}_{\tilde{i}\tilde{j}k}$	Expected reward of a transition states $\tilde{i}$ to $\tilde{j}$ under action a and scenario k ($\widetilde{\widetilde{\boldsymbol{r}}}^{a}_{k}$ matrix form)
$\widetilde{\widetilde{o}}^{a}_{\tilde{i}O}$	Probability of observing O when the equivalent PODMP state is $\tilde{i}$ under action a
g	The number of reward levels

Variables:

$\widetilde{Y}_t$	Observed compound state
$\widetilde{\widetilde{Y}}_t$	Unobserved "pure" state of system in period t (combination of observable and scenario k)
$b_{t+1}(i, a, o)$	Belief state probability in period t is in state i after action a and observation o
$b_t(i, a_{t-1}, o_{t-1})$	Belief state probability in period t is in state i after hypothetical action a_{t-1} and observation o_{t-1}
G	The reward level which is a discretized version of $r^{\pi(O_{11}...O_{tW})}_{Y_{(t-1)W}Y_{tw}} + \epsilon_{tw}$
$\widetilde{\alpha}^{l}_{t}(i, a, o)$	The component i of the l element of a set for action a and o observation in period t
$\alpha^{l}_{t}(i)$	The component i of the l element of the set of putatively optimal policy vectors in period t

The sparsity of the equivalent POMDP transition $\widetilde{\widetilde{\boldsymbol{p}}}^{a}_{e}$ and $\widetilde{\widetilde{\boldsymbol{r}}}^{a}_{e}$ expected reward matrices relates to the assumption that only one of the scenario correctly describes the true system. Therefore, transitions from pure states related to one scenario to those relating to other scenarios are impossible. Given an observation of the natural state, the belief state must reside within a sub-region as described in Hou (2015). The details of calculating observation matrices for the POMDP calculation inputs are described in the next section.

3.3. Reward level-based learning

Next, we derive the observation matrix $O_{tw} = \{\widetilde{Y}_t, G\}$. With g reward levels, there are Ng possible observations. Further, assume that the random reward deviations, ϵ, are normally distributed with mean zero and known standard deviation, σ. Relevant reward level cutoff values L_g can be derived using:

$$L_G = \min_{a,\tilde{i}\tilde{j},k}(\widetilde{\widetilde{r}}^{a}_{\tilde{i}\tilde{j}k} - 3\sigma) + (G-1)[\max_{a,\tilde{i}\tilde{j},k}(\widetilde{\widetilde{r}}^{a}_{\tilde{i}\tilde{j}k}) - \min_{a,\tilde{i}\tilde{j},k}(\widetilde{\widetilde{r}}^{a}_{\tilde{i}\tilde{j}k}) + 6\sigma]/g \quad \forall\, G = 1, \ldots, g+1. \tag{7}$$

Then, the POMDP equivalent "observation matrix" ($\widetilde{\widetilde{o}}^{a}_{io}$) gives the probability of observing $O_{tw} = \{Y_{t+1w}(k, a), R\}$ when the true pure state is i under action a is given by:

$$\widetilde{\widetilde{o}}^{a}_{io} = p^{a}_{i_o j_o k_o}\,[\Phi(L_{l_o+1}, r^{a}_{i_o j_o k_o}, \sigma) - \Phi(L_{l_o}, r^{a}_{i_o j_o k_o}, \sigma)] \quad \forall\, a = 1, \ldots, u;\ i = 1, \ldots, Nq;\ o = 1, \ldots, Ng \tag{8}$$

and $i_o = ((i-1) mod\ N) + 1$, $j_o = ((o-1) mod\ N) + 1$,

$$k_o = \frac{i-1}{N} + 1,\ l_o = \left\lfloor \frac{o-1}{N} \right\rfloor + 1 \tag{9}$$

and where $\Phi()$ is the cumulative normal distribution and $\lfloor\ \rfloor$ is the round-down operation.

Eq. (12) shows the conversion from pure states to natural: from state i_o, natural to state j_o, and from scenario k_o, to scenario l_o. Because the number of reward levels, g, can be large and the standard deviation, σ, can be small, each observed reward level can be relatively informative compared with single transition observations. Therefore, the use of reward levels to gain information is a relatively fast process. The relative speeds are quantified using numerical examples in Section 5.

As an example, assume $W = 1$, $N = 2$ states, $A = 2$ actions, $q = 2$ scenarios, $\sigma = 1$, and $g = 4$ levels.

p^1_{111}= 0.7	p^1_{121}= 0.3	r^1_{111}=5	r^1_{121}= 5	p^2_{111}= 0.5	p^2_{121}= 0.5	r^2_{111}=4	r^2_{121}= 4
p^1_{211}= 0.6	p^1_{221}=0.4	r^1_{211}=0	r^1_{211}=0	p^2_{211}= 0.5	p^2_{221}=0.5	r^2_{211}=-1	r^2_{211}=-1
p^1_{112}= 0.7	p^1_{122}= 0.3	r^1_{112}=5	r^1_{122}=5	p^2_{112}= 0.9	p^2_{122}= 0.1	r^2_{112}=4	r^2_{122}=4
p^1_{212}= 0.6	p^1_{222}= 0.4	r^1_{212}=0	r^1_{222}=0	p^2_{212}=0.8	p^2_{222}= 0.2	r^2_{212}=1	r^2_{222}=1

The cutoff values from equation (11) are $L_1 = -4$, $L_2 = -1$, $L_3 = 2$, $L_4 = 5$, $L_5 = 8$. Then, the pure states are combinations indexed by i as shown on the left-hand-side of Table 4. The observations are combinations of the "to" states and the reward level as indicated by the top of Table 4. The observation matrix from Eq. (11) is given in Table 4 for action $a = 2$.

3.4. Perseus POMDP point-based solver

Exact optimal POMDP solution methods may prove to be computationally expensive and difficult to implement for large problems and infinite horizons. At the same time, only a small subset of feasible belief points are "reachable" states, i.e., likely to be encountered in practice. Methods such as Point Based Value Iteration (PBVI) involve only reachable points which occupy only a portion of a full simplex (Pineau, Gordon, Thrun, 2003, Shani, 2006). These point-based methods identify reachable points and steps or "trajectories" from these points. Then, the methods attempt to solve approximately the steady state or "Bellman" equations (Bellman, 1957) at these points in reasonable computational time. "PERSEUS" is one such PBVI developed by Spaan and Vlassis (2005) with desirable computational cost and accuracy tradeoff.

Specifically, PERSEUS generates and updates the set of non-optimal alpha vectors through iterations ("backup" stages) of value assignments at the belief points. At each backup stage, the value of each point in the belief set is improved or stays the same. An innovation associated with these PERSEUS algorithm is that a single backup operation may improve the values of many points, thereby speeding up the solution process for large problems.

In our description, there is an assumption that the reward $\widetilde{\widetilde{r}}^a_{\widetilde{i}\,\widetilde{\widetilde{j}}}$ depends only on the "to" state $\widetilde{\widetilde{j}}$ and action $a \in A$. The PERSEUS method involves the marginal probability that an observation is derived given an action and a belief state:

$$\Pr\left\{o|a,\, b_t(1,\, a_{t-1},\, o_{t-1}),\ldots,b_t\left(\widetilde{\widetilde{N}},\, a_{t-1},\, o_{t-1}\right)\right\} = \sum_{l=1}^{\widetilde{\widetilde{N}}} o^a_{lo} \sum_{i=1}^{\widetilde{\widetilde{N}}} \widetilde{\widetilde{p}}^a_{il}\, b_t(i,\, a_{t-1},\, o_{t-1}) \quad \forall\, o = 1,\, \ldots,Ng \tag{10}$$

which drives from the probability that the system will be in the state l and the observation will result. The next period belief state after an action a and an observation o is taken derives from Bayes theorem:

$$b_{t+1}(j,\, a,\, o) = \frac{o^a_{jo} \sum_{i=1}^{\widetilde{\widetilde{N}}} \widetilde{\widetilde{p}}^a_{ij}\, b_t(i,\, a_{t-1},\, o_{t-1})}{\sum_{l=1}^{\widetilde{\widetilde{N}}} o^a_{lo} \sum_{i=1}^{\widetilde{\widetilde{N}}} \widetilde{\widetilde{p}}^a_{il}\, b_t(i,\, a_{t-1},\, o_{t-1})} \quad \forall\, a = 1,\, \ldots,u;\, j = 1,\, \ldots,\widetilde{\widetilde{N}}. \tag{11}$$

Table 3
Description of the BRL formulation and the corresponding POMDP formulation.

Parameter or Variable	Compound System BRL Formulation	Equivalent POMDP
States	$\widetilde{Y}_t = 1,\, \ldots,\widetilde{N}$ (Combinations of n_{it} $\forall\, i = 1,\, \ldots,N$)	$\widetilde{\widetilde{Y}}_t = 1,\, \ldots,\widetilde{\widetilde{N}}$ with $\widetilde{\widetilde{N}} = Nq$ (Combinations of n_{it} $\forall\, i = 1,\, \ldots,N$ and k)
Starting Conditions	n_{it} $\forall\, i = 1,\, \ldots,N$ & $p_t(k)$ $\forall\, k = 1,\, \ldots,q$	$\boldsymbol{b}_t = \begin{cases} p_t(k) & \forall\, k = 1,\, \ldots,q \\ 0 \end{cases}$
Policy	$\pi(O_{11}\ldots O_{tW})$	$\pi(O_{11}\ldots O_{tW})$
Transition Probabilities	$\widetilde{\boldsymbol{p}}^a_k$ $\forall\, k = 1,\, \ldots,q$ $\forall\, a = 1,\, \ldots,u$	$\widetilde{\widetilde{\boldsymbol{p}}}^a_e = \begin{pmatrix} \widetilde{\boldsymbol{p}}^a_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \widetilde{\boldsymbol{p}}^a_q \end{pmatrix}$ $\forall\, a = 1,\, \ldots,u$
Expected rewards	$\widetilde{\boldsymbol{r}}^a_k$ $\forall\, k = 1,\, \ldots,q$ $\forall\, a = 1,\, \ldots,u$	$\widetilde{\widetilde{\boldsymbol{r}}}^a_e = \begin{pmatrix} \widetilde{\boldsymbol{r}}^a_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \widetilde{\boldsymbol{r}}^a_q \end{pmatrix}$ $\forall\, a = 1,\, \ldots,u$
Observations	The compound state is observable, $\widetilde{Y}_t$	The generalized state, $\widetilde{\widetilde{Y}}_t$, is not observable, only the compound state, $\widetilde{Y}_t$, and the reward level G
Observation Matrix	This is not an input.	$o^a_{\widetilde{i}\,O}$ $\forall\, a = 1,\, \ldots,u; \widetilde{\widetilde{i}} = 1,\, \ldots,\widetilde{\widetilde{N}}$

Table 4
The $\boldsymbol{a} = 2$ observation matrix for Example 1 from Eq. (4).

			Observation (o)	1	2	3	4	5	6	7	8
			To state (j_o)	1	2	1	2	1	2	1	2
Pure state	From state	Scenario	Reward level (G)	1	1	2	2	3	3	4	4
(i)	(i_o)	(k_o)					o^a_{io}				
1	1	1		0.000	0.000	0.020	0.002	0.737	0.082	0.143	0.016
2	2	1		0.299	0.199	0.299	0.199	0.001	0.001	0.000	0.000
3	1	2		0.000	0.000	0.020	0.002	0.737	0.082	0.143	0.016
4	2	2		0.018	0.005	0.655	0.164	0.127	0.032	0.000	0.000

Note that the denominator in Eq. (11) is the marginal probability in Eq. (10). Also, consider the construction of the belief state implied by Table 3. Then, for our Bayesian Reinforcement Learning (BLR) problem, the values of update probabilities are the scenario probabilities,

$$p_t(k, O_{11}...O_{tW}) = b_t(\widetilde{Y}_t + Nk, a, o) \quad \forall\, k = 1, ..., q. \tag{12}$$

Smallwood and Sondik (1973) discovered that the optimal value function for POMDP problems is piecewise linear in the belief space probabilities and the control policy vector

$$V_t\left(b_{t+1}(1, a, o), ..., b_{t+1}\left(\widetilde{\widetilde{N}}, a, o\right)\right) = a \text{ from argmax}_{\{[\alpha_t^l(1),...,\alpha_t^l(\widetilde{\widetilde{N}})]_l} \left[\sum_{i=1}^{\widetilde{\widetilde{N}}} \alpha_t^l(i) b_t(i, a_{t-1}, o_{t-1})\right]. \tag{13}$$

$$backup\left(\left(b_t(1, a_{t-1}, o_{t-1}), ..., b_t\left(\widetilde{\widetilde{N}}, a_{t-1}, o_{t-1}\right)\right)\right) = argmax_{\{[\widetilde{\alpha}_t^l(1,a,o),...]\}_a} \left\{\sum_{i=1}^{\widetilde{\widetilde{N}}} b_t(i, ...)\left[\widetilde{\widetilde{r}}_{i1}^a + argmax_{\{[\widetilde{\alpha}_t^l(1,a,o),...]\}_l} \sum_{i=1}^{\widetilde{\widetilde{N}}} b_t(i, ...)\widetilde{\alpha}_t^l(i, a, o)\right]\right\} \tag{15}$$

where $b_t(i, ...) = b_t(i, a_{t-1}, o_{t-1})$. The steps of the PERSEUS algorithm are given below.

Algorithm: PERSEUS from Spaan and Vlassis (2005)

Generate Belief-points Generate a set of reachable belief-points $\boldsymbol{B}$ using simulation.

Initialize Set $\vartheta_{t+1} = \varnothing$, $\widehat{\boldsymbol{B}} = \boldsymbol{B}$.

Sample Belief-point $\left(b_t(1, a_{t-1}, o_{t-1}), ..., b_t\left(\widetilde{\widetilde{N}}, a_{t-1}, o_{t-1}\right)\right)$ from $\widehat{\boldsymbol{B}}$ uniformly at random.

Backup Calculate $(\alpha_1, ..., \alpha_{\widetilde{\widetilde{N}}})' = backup\left(\left(b_t(1, a_{t-1}, o_{t-1}), ..., b_t\left(\widetilde{\widetilde{N}}, a_{t-1}, o_{t-1}\right)\right)\right)$.

Update If $\sum_{i=1}^{\widetilde{\widetilde{N}}} b_t(i, a_{t-1}, o_{t-1})\alpha(i) < V_t\left(b_t(1, a_{t-1}, o_{t-1}), ..., b_t\left(\widetilde{\widetilde{N}}, a_{t-1}, o_{t-1}\right)\right)$

include $(\alpha_1, ..., \alpha_{\widetilde{\widetilde{N}}})'$= to ϑ_{t+1}, else include

$$(\alpha_1, ..., \alpha_{\widetilde{\widetilde{N}}})' = argmax_{\left\{\left[\widetilde{\alpha}_t^l(1,a,o),..., \left[\widetilde{\alpha}_t^l\left(\widetilde{\widetilde{N}}, a, o\right)\right]\right]\right\}_l} \sum_{i=1}^{\widetilde{\widetilde{N}}} b_t(i, a_{t-1}, o_{t-1})\alpha_t^l(i)\ .$$

Filter Update $\widehat{\boldsymbol{B}}$ with unimproved points

$\widehat{\boldsymbol{B}} = \{b \in \boldsymbol{B}, V_{t+1}(b_t(1, a_{t-1}, o_{t-1}), ...,) < V_t(b_t(1, a_{t-1}, o_{t-1}), ...,)\}$.

Check If $\widehat{\boldsymbol{B}}$ is empty, **STOP** else go to **Sample**

Writing a single step optimization and including the assumption that the system is in steady state gives

$$V_{t+1}\left(b_t(1, a_{t-1}, o_{t-1}), ..., b_t\left(\widetilde{\widetilde{N}}, a_{t-1}, o_{t-1}\right)\right) = \max_a \left[\sum_{i=1}^{\widetilde{\widetilde{N}}} b_t(i, a_{t-1}, o_{t-1})\widetilde{\widetilde{r}}_{i1}^a + \gamma \sum_{o=1}^{Ng} \Pr\{o|a, b_t(1, a_{t-1}, o_{t-1}), ...\} V_t(b_{t+1}(1, a, o), ...\right]$$

$$= \max_a \left[\sum_{i=1}^{\widetilde{\widetilde{N}}} b_t(i, a_{t-1}, o_{t-1})\widetilde{\widetilde{r}}_{i1}^a + \gamma \sum_{o=1}^{Ng} \underset{\{[\widetilde{\alpha}_t^l(1),...,\widetilde{\alpha}_t^l(\widetilde{\widetilde{N}})]\}_l}{\text{Minimize}} \left[\sum_{i=1}^{\widetilde{\widetilde{N}}} b_t(i, a_{t-1}, o_{t-1})\widetilde{\alpha}_t^l(i, a, o)\right]\right]. \tag{14}$$

where

$$\widetilde{\alpha}_t^l(i, a, o) = \sum_{j=1}^{\widetilde{\widetilde{N}}} \Pr\left\{o|a, b_t(1, a_{t-1}, o_{t-1}), ..., b_t\left(\widetilde{\widetilde{N}}, a_{t-1}, o_{t-1}\right)\right\} \widetilde{\widetilde{p}}_{ij}^a \alpha_t^l(i).$$

The backup vector operation is intended to generate belief-point b is given by

In our numerical studies, the initial belief set is generated using a random walk from sampled compound states. PERSEUS then samples a single belief-point from $\widehat{\boldsymbol{B}}$, generates an alpha vector using a backup for this point and then checks adds the alpha vector to the set ϑ_{t+1} if there is an associated improvement. The values of the belief points with the new alpha vectors are continually improved with respect to achieving equality in the steady state equations. The steps are repeated until all the points in $\widehat{\boldsymbol{B}}$ reach steady state to a good approximation.

4. Rigorous results

Duff (2002) proposed the idea to treat the problem in Eqs. (1), (2), and (3) approximately as a POMDP as he focused on the $q = \infty$ scenario case for a single system, $W = 1$. Then, Wang, Won, Hsu, and Lee (2012) considered the finite scenario $q \ll \infty$ single system case also for a single system. Wang, Won, Hsu, and Lee (2012) noted that the problem is a POMDP as indicated in Table 3 and used a point-based solver to generate putatively optimal policy parameters, $\alpha_{t=\infty}^l(i)$, for steady state and $i = 1, ..., \widetilde{\widetilde{N}}$.

The results here formalize the correspondence in Table 3 which Wang, Won, Hsu, and Lee (2012) and others assumed and establish that correspondence for the case, $W > 1$. Benefits for achieving these results including verifying the formulas in Table 3. Also, the results potentially aid in training BRL. Also, Wang, Won, Hsu, and Lee (2012) focused on approximating the $q = \infty$ case with successive sets of scenarios. Here, we argue that finite numbers of scenarios are sufficient to generate valuable insights for our cyber security maintenance case study.

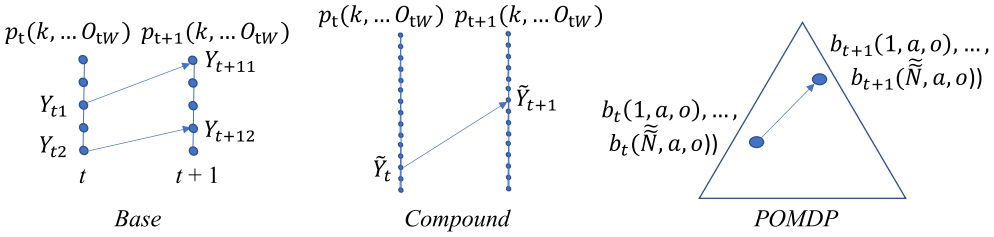

Fig. 3. Arbitrary transition in the three equivalent systems.

Theorem 1. Consider three BRL formulation in Eqs. (1), (2), and (3) formulations. For all formulations, the control systems have learning available about the scenarios through Eqs. (11) and (12). Also, assume that there is only a single reward level, $g = 1$. The three formulations are the:

a. (Base) $W > 1$ problem.
b. (Compound) $W = 1$ Exact $\widetilde{N}$, $\widetilde{\boldsymbol{p}}_k^a$, and $\widetilde{\boldsymbol{r}}_k^a$ inserted for N, $\boldsymbol{p}_k^a$, and $\boldsymbol{r}_k^a$ $\forall\ k = 1, \ldots, q \quad \forall\ a = 1, \ldots, u$.
c. (POMDP) $W = 1$ and $\widetilde{\tilde{N}}$, $\widetilde{\tilde{\boldsymbol{p}}}_k^a$, and $\widetilde{\tilde{\boldsymbol{r}}}_k^a$ from Table 3 (right-hand-side) $\forall\ k = 1, \ldots, q \quad \forall\ a = 1, \ldots, u$.

All three problems are equivalent. The decision spaces are the same. An arbitrary policy $\pi^*(O_{11}\ldots O_{tW})$ has the same discounted cost in all policies. Also, an optimal control policy, $\pi^*(O_{11}\ldots O_{tW})$, for one problem is optimal for all problems.

Proof. We begin with some preliminary identifications. Then, we establish the corresponding period transitions for the three formulations. Given equivalent policies, the probabilities of each of these transitions are found to be the same and the expected rewards are the same conditioned on the transitions. Since the transition probabilities and expected rewards are the same, the objective function values for each policy are the same so the formulations are equivalent.

First consider that the assignment of action combinations to subsystems in the same state for a compound system actions has no effect on the transition probabilities or expected rewards. Therefore, without loss of generality, the index of the first subsystem in each state can be assigned a lowest action number. Therefore also, all policies have the same effective inputs (observed natural states for subsystems or the joint system) and outputs (action assignments to subsystems).

Next, consider an arbitrary transition in the base system as indicated in Fig. 3. The transition is from one identifiable state to any one of an equivalent set of states with the same counts. The subsystem transitions and the scenario probabilities are updated using Bayes theorem. The corresponding transition for the compound system (also shown in Fig. 3) does not account for the subsystem identities without loss of generality. Again, Bayes theorem is used. Similarly, with the same observations, including the compound system, the belief states in the equivalent POMDP move as dictated by Bayes' theorem.

Assume that the action is a. Consider that the equivalent transitions from the base and compound systems have the same probabilities and expected reward as established by Eqs. (11) and (13). The equivalent compound system transition probability is

$$\Pr\{\widetilde{Y}_{t+1}|\widetilde{Y}_t, O_{11}\ldots O_{tW}\} = \sum_{k=1}^{q} p_t(k, O_{11}\ldots O_{tW})\widetilde{p}^a_{\widetilde{i}\widetilde{j}\,k} \tag{16}$$

Similarly, for the equivalent POMDP system, the marginal transition probability is

$$\begin{aligned}\Pr\{o \in \widetilde{Y}_{t+1}|\widetilde{Y}_t, O_{11}\ldots O_{tW}\} &= \sum_{l=1}^{\widetilde{\tilde{N}}} o^a_{lo} \sum_{i=1}^{\widetilde{\tilde{N}}} \widetilde{\tilde{p}}^a_{il}\, b_t(i, a_{t-1}, o_{t-1}) \\ &= \sum_{i=1}^{\widetilde{\tilde{N}}} \widetilde{\tilde{p}}^a_{il}\, b_t(i, a_{t-1}, o_{t-1}) = \sum_{k=1}^{q} p_t(k, O_{11}\ldots O_{tW})\widetilde{\tilde{p}}^a_{\widetilde{i}\widetilde{j}\,k}\end{aligned} \tag{17}$$

where $o^a_{lo} = 1$ because $g = 1$ for $o \in \widetilde{Y}_{t+1}$ or 0 otherwise and $b_t(i, a_{t-1}, o_{t-1}) = b_t(\widetilde{Y}_t + Nk, a, o) \quad \forall\ k = 1, \ldots, q$ or 0 otherwise. Similarly, the expected

$$\begin{aligned}\underset{Y_{11}(k,a),\ldots Y_{HQ}(k,a)}{\mathrm{E}}\left[\widetilde{r}^a_{\widetilde{Y}_t\widetilde{Y}_{t+1}k} + |\widetilde{Y}_t\right] &= \sum_{k=1}^{q} p_t(k, O_{11}\ldots O_{tW}) \underset{Y_1,\ldots Y_H}{\mathrm{E}}\left[\left(r^{\pi(O_{11}\ldots O_{tW})}_{\widetilde{Y}_{(t-1)}\widetilde{Y}_t} + \epsilon\right)\right]. \\ &= \sum_{k=1}^{q} p_t(k, O_{11}\ldots O_{tW})\widetilde{r}^a_{\widetilde{Y}_t\widetilde{Y}_{t+1}k}.\end{aligned} \tag{18}$$

For the POMDP we have

$$\begin{aligned}\underset{Y_{11}(k,a),\ldots Y_{HQ}(k,a)}{\mathrm{E}}[\widetilde{\tilde{r}}^a_{\widetilde{\tilde{Y}}_t\widetilde{\tilde{Y}}_{t+1}} + |\widetilde{\tilde{Y}}_t] &= \sum_{i=1}^{\widetilde{\tilde{N}}} b_t(i, a_{t-1}, o_{t-1}) r^a_{\widetilde{\tilde{Y}}_t\widetilde{\tilde{Y}}_{t+1}} \\ &= \sum_{k=1}^{q} p_t(k, O_{11}\ldots O_{tW})\widetilde{\tilde{r}}^a_{\widetilde{\tilde{Y}}_t\widetilde{\tilde{Y}}_{t+1}k}.\end{aligned} \tag{19}$$

Therefore, for a given action, the equivalent steps have the same probability and associated expected reward. This holds for every epoch if the same updates are used which maintain the relationship between the scenario probabilities.

Also, it follows that each possible sample path has the same probability in both formulations. Since the overall objective value for both problems is a sum over the same expected rewards, the two formulations are equivalent. Because of the one-to-one correspondence of policy solutions and objective values, the same solutions are optimal in both formulations. □

This proof establishes the equivalence of the three formulations. It also clarifies the fact that there are many details relating to this equivalence. These include the assumption of a single reward level, $g = 1$. We feel that this assumption can be generalized with future research.

Another result relates to the attainment of Bellman conditions in Eq. (14) at one or more points in the belief space. We define "pure" beliefs in the context of POMDP as having all the scenarios combined into a single matrix formulation. For pure beliefs, the scenario is known, and the solution is, therefore, an ordinary Markov decision process (MDP) policy.

Theorem 2. It is possible to achieve equality in equation (14) at a mixed belief point.

Proof. In run 7 of the computational experiment in the next section, exact equality is achieved at multiple belief points. In fact, equality to within 0.0005% at over 10,000 belief points is achieved. □

Remark. In all the numerical examples, the differences between the left hand and right-hand sides of equation (14) for the putative solutions

Table 5
Transition counts from hypothetical numerical data and (b) expected rewards. Parameters with non-negligible parametric uncertainty are underlined. The 0.25 count added is a smoothing parameter.

(a)			(b)	
Action #1	State 1	State 2	Action #1 Mean Reward	Standard Deviation Mean Reward
State 1	8000	2000	−1000.0001	0
State 2	8000	2000	1000	0
Action #2	State 1	State 2	Action #2 Mean Reward	Standard Deviation Mean Reward
State 1	8.25	2.25	-1000	1000
State 2	8.25	2.25	1000	1000

averaged over 5,000 belief points spread out within the feasible region as pictured in Fig. 1(b).

Attaining steady state conditions (at least approximately) is remarkable because in all the cases the control system is learning. While virtually all previous research has assumed that steady state conditions were attainable, it might be natural to expect that the system is transitioning while learning is occurring. In many of the cases, the system is rapidly eliminating parametric uncertainty in only a few steps.

5. Numerical study

In this section, we introduce a measure of the speed that the derived control system learns called the "median estimated learning time" (MELT). This is similar to the Average Learning Time (ALT) described in a recent dissertation omitted for confidentiality, but the MELT is much less computationally intensive for estimation since medians are not influenced by rare tail events with large values than averages. Therefore, fewer replicates are needed to estimate the expected median compared with the mean. Then, we compare control systems from BRL formulations with different assumptions. This includes with and without OALH selection of scenarios, multiple identical systems, and multiple reward levels. Additionally, nominal, robust, random, epsilon greedy, and feature-based policies are studied.

5.1. Median estimate learning time

For many control strategies, the Bayesian updating in Eqs. (11) and (12) converges to eliminate the parametric uncertainty. Then, the system converges to a pure state, i.e., $p_{t=\infty}(k, O_{11}...O_{t=\infty W}) \approx 1$, for the ground truth scenario and is near zero for all other k values. Yet, this may require thousands or even an infinite number of periods to occur. Therefore, the median number of periods is used for estimation. The additional required definitions follow:

Parameters:

$N_{simulated}$ The number of simulation runs used for evaluation.

Variables:

$L_{(i)}$ The i^{th} highest number of periods until at least one pure state probability exceeds 0.5.

For efficient estimation, our proposed Monte Carlo-based simulation method is based on the Harrell and Davis (1982) quantile estimation method. Then, the proposed measure of efficiency is the median estimated learning time (MELT):

$$MELT = \phi_{HD}[L_{(1)}, ..., L_{(N_{simulated})}] = \sum_{i=1}^{N_{simulated}} w_{N_{simulated},(i)} L_{(i)} \quad (20)$$

where

$$w_{N_{simulated},(i)} = I_{i/N_{simulated}}\{0.5(N_{simulated}+1), 0.5(N_{simulated}+1)\} - I_{(i-1)/N_{simulated}}\{0.5(N_{simulated}+1), 0.5(N_{simulated}+1)\}.$$

where $I_c(a, b)$ denotes the incomplete beta function.

In our numerical study, we use 10,000 Monte Carlo simulations involving 100 randomly chosen starting points and with each simulation truncated at 1000 steps. Therefore, our MELT is technically a lower bound with 1000 included if the simulation has not terminated. The MELT construction means that the truncation has generally negligible effects on the estimate.

5.2. Problem setup

The numerical study involves two actions and two states. Action #1 is thoroughly studied and well known because it is in use. The number of transitions for Action #1 are shown in Table 5(a). The expected reward parameters are given in Table 5(b). These counts are sufficiently large that parametric uncertainty can be ignored. Action #2 is less studied with only 10 total transitions known for each state. The standard deviation for the mean rewards are also high because of the limited data. The parameters with parametric uncertainty are shown in Table 5 with underlines.

All sixteen runs in our experiment are based on the data in Table 5. In some sense, the runs are all solving the same infinite scenario problem. Yet, in each case the model scenarios are sampled so that each run corresponds to a separate formulation in equation (1). For all problems, the nominal policy is based on a naïve estimate of the probabilities and the expected rewards and Markov decision processes (MDP). Therefore, the nominal policy is to use Action #1 in all cases. The discount factor is $\gamma = 0.95$.

Our implementation of PERSEUS by Walraven (2017). Solution times are approximately twenty minutes using an i7 seventh generation processor. All codes terminated when the root mean squared absolute differences for the two sides of Eq. (14) were less than 0.1% of the objective value or less. For example, Table 6 shows the solution "alpha" vectors for Run 1. To apply this solution, the point in belief space is updated based on the most recent observation and the past point using Bayes' theorem in Eq. (11). Then, the belief points are each multiplied by an alpha vector and the highest product is the action recommended by the policy. All optimal

Table 6
The putative optimal policy for Run 1 is characterized by these alpha vectors. The belief state probabilities are multiplied by the values and then the action taken is at left associated with the highest product.

2	−35157.9	−33501.0	−8655.7	−6680.9	19884.6	21447.6	−11110.5	−8124.8	−6016.5	−3915.7	−24064.9	−22006.7
2	−34235.6	−32592.8	−8695.8	−6721.0	12884.6	14515.7	−11301.8	−8339.2	−6026.8	−3926.0	−23989.6	−21931.4
1	−31894.3	−29894.3	−8984.4	−6984.4	8737.3	10737.3	−11355.8	−9355.8	−6390.5	−4390.5	−23095.6	−21095.6
1	−31312.4	−29312.4	−9057.0	−7057.0	7806.3	9806.3	−11431.2	−9431.2	−6456.2	−4456.2	−22926.5	−20926.5
2	−33829.3	−32095.5	−8722.5	−6747.8	11475.2	12794.9	−11364.4	−8451.3	−6037.4	−3937.0	−23931.9	−21873.2
1	−31312.4	−29312.4	−9057.0	−7057.0	7806.3	9806.3	−11431.2	−9431.2	−6456.2	−4456.2	−22926.5	−20926.5

Table 7
The full factorial experimental plan for the numerical study.

Run	Scenarios	Type	# Subsystems	# Levels	Nominal		Random	
	(q)		(W)	(g)	Value	MELT	Value	MELT
1	6	LHS	1	1	−11579.7	> 1000	−11057.6	13.6
2	6	LHS	1	10	−11555.4	> 1000	−11261.6	2.0
3	6	random	1	1	−11630.0	> 1000	−12745.6	46.9
4	6	random	1	10	−11352.3	> 1000	−5171.0	2.0
5	6	LHS	2	1	−11482.4	> 1000	−9853.9	127.4
6	6	LHS	2	10	−11486.7	> 1000	−10740.1	129.2
7	6	random	2	1	−11447.0	> 1000	−16682.9	204.1
8	6	random	2	10	−11528.8	> 1000	−13557.9	24.5
9	12	LHS	1	1	−11520.0	> 1000	−12580.8	319.2
10	12	LHS	1	10	−11456.4	> 1000	−12526.0	5.6
11	12	random	1	1	−11580.2	> 1000	−12045.1	299.3
12	12	random	1	10	−11347.1	> 1000	−10496.9	5.5
13	12	LHS	2	1	−11603.8	> 1000	−10672.3	905.4
14	12	LHS	2	10	−11442.5	> 1000	−9721.8	5.0
15	12	random	2	1	−11578.6	> 1000	−10044.3	630.6
16	12	random	2	10	−11570.0	> 1000	−10598.2	5.9

Table 8
The full factorial experimental outputs from the numerical study.

Run	Epsilon-Greedy 0.90		Epsilon-Greedy 0.95		Feature-Based		Optimal	
	Value	MELT	Value	MELT	Value	MELT	Value	MELT
1	−12433.2	> 1000	−12891.4	> 1000	−11579.7	> 1000	−12934.6	> 1000
2	−15868.4	17.2	−16170.1	23.4	−5791.5	201.9	−16909	29.9
3	−10045	> 1000	−10748.4	> 1000	−11630	> 1000	−15554	> 1000
4	−11044.5	64.9	−11698.1	126.3	−11484.4	> 1000	−11988	> 1000
5	−24922.8	> 1000	−24460	> 1000	−11482.4	> 1000	−24918.3	> 1000
6	−32854	16.4	−33961.5	23.2	−21582.6	236.5	−35146.7	47.82
7	−33401.4	> 1000	−34666.1	> 1000	−11447	> 1000	−35026.7	> 1000
8	−22955	45.7	−23927.8	74.3	−23200.3	> 1000	−24755	> 1000
9	−13330.5	> 1000	−12068.3	> 1000	−11520	> 1000	−12739.8	> 1000
10	−16663.6	46.2	−16452.5	61.8	−11823.7	> 1000	−16763.6	> 1000
11	−17815.5	> 1000	−17179.7	> 1000	−11580.2	> 1000	−17171	> 1000
12	−16377.2	121.8	−16591.8	187.2	−10167.8	62.2	−16326.5	515.1
13	−28997.4	> 1000	−30210.4	> 1000	−11603.8	> 1000	−29184.4	> 1000
14	−35483.6	27.2	−37501.2	35.6	−19088	4.0	−36744	> 1000
15	−21084.5	> 1000	−21734.7	> 1000	−11578.6	> 1000	−22773.6	> 1000
16	−35501.1	101	−35523.3	209.7	−25491.1	> 1000	−35439.6	> 1000

policies include a mixture of actions 1 and 2 so that the recommendations are opportunistic. As the problem ground truth scenario is learned, the system converges to the appropriate actions.

The epsilon greedy policy takes the putatively optimal actions (from the alpha vectors) with a high probability and a random action with a low probability. For example, epsilon greedy with 0.95 (E95) takes the optimal with 95% probability and a random action with 5% probability. The feature-based learning that we explore only observes the reward level and not the state transition in the related POMDP. Therefore, for cases with a single level of observed rewards, feature-based learning is equivalent to nominal action taking.

5.3. Experiment plan and results

The sixteen runs in the numerical study differ by four factors: Scenarios (q), Scenario Type (Random or LHS), Number of Subsystems (W), and Number of Learning Levels (g). The two-level full factorial is shown in Tables 7 and 8 together with the results from the numerical study for the estimated discounted expected rewards (values) and the median estimated learning time (MELT) values.

For both responses, the analysis of variance (ANOVA) results are significant with p-value less than 0.001. The p-values for the specific

Table 9
The p-values from an analysis of variance for second order models are provided.

Source	Value P-Value	MELT P-Value
Model	0.001	0.000
Linear	0.000	0.000
Scenarios	0.302	0.005
Type	0.534	0.282
Subsystems	0.267	0.446
Learning levels	0.812	0.013
Method	0.000	0.000
2-Way Interactions	0.120	0.039
Scenarios*Type	0.630	0.388
Scenarios*Systems	0.024	0.767
Scenarios*Learning levels	0.297	0.038
Scenarios*Method	0.559	0.053
Type*Systems	0.157	0.231
Type*Learning levels	0.104	0.023
Type*Method	0.887	0.177
Systems*Learning levels	0.275	0.117
Systems*Method	0.643	0.545
Learning levels*Method	0.031	0.190

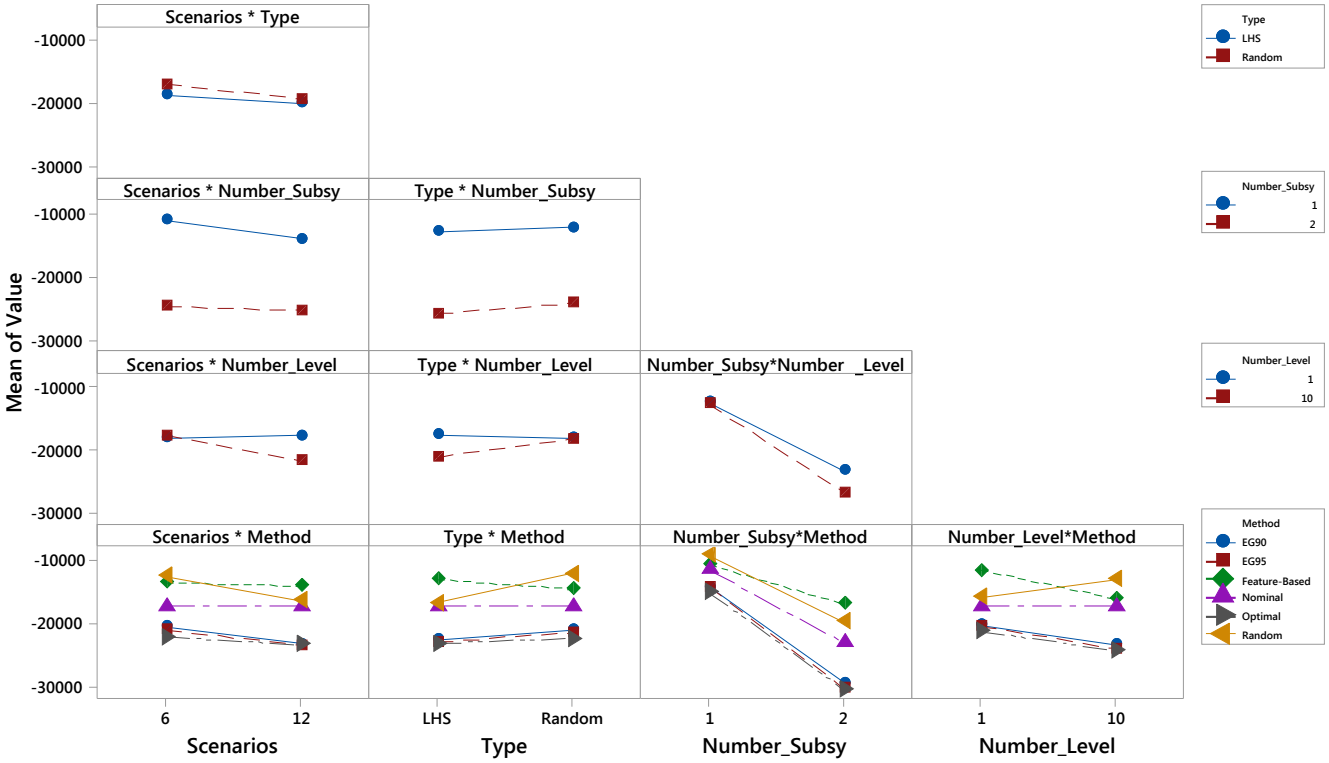

Fig. 4. Interaction plots for the discounted expected reward per subsystem (value/W).

terms are shown in Table 9. The linear model predictions for the expected discounted rewards (values) and the median estimated learning time (MELT) values are shown in Fig. 4 and Fig. 5 respectively.

Findings from inspecting the interaction plots (Fig. 4 and Fig. 5) and the p-values (Table 8) are listed by position in the plots starting in the upper-left and going down each row. These include:

The Latin hypercube (LHS) formulates result in marginally more consistent performance as the number of scenarios changes because the methods approximate better infinity scenarios.

Having additional subsystems pooling information offers an improved objective value (lower) per host. The effect is reduced by a change in the numbers of scenarios.

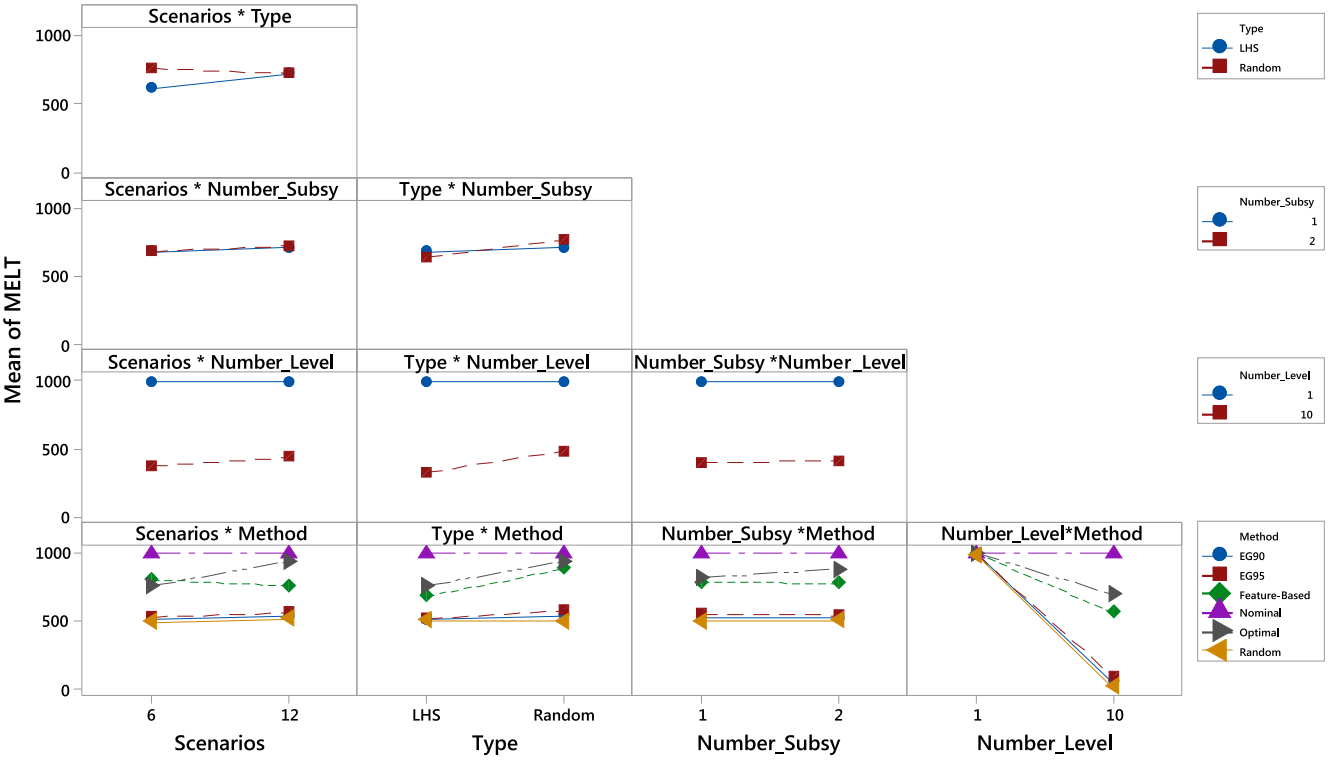

Fig. 5. Interaction plots for the median estimated learning time (MELT) values.

Table 10
Actual month transition counts for Linux critical hosts and monthly direct costs for three actions. The underlined quantities indicated uncertainty in associated parameters and include smoothing 0.1.

Action #1: Limited Effort	State 1	State 2	State 3	State 4	Est. Mean Reward	Stdev. Mean Reward
State 1	16,075	1404	35	15	7.0	0
State 2	412	127,519	756	59	10.6	0
State 3	0	0	1	0	17.7	0
State 4	0	0	0	1	3000.0	1000
Action #2: Research Accept	**State 1**	**State 2**	**State 3**	**State 4**	**Est. Mean Reward**	**Stdev. Mean Reward**
State 1	1	0	0	0	59.4	0
State 2	1	0	0	0	63.1	0
State 3	34	429	610	4	70.2	0
State 4	8	53	18	799	103.7	20
Action #3: Compensating Controls	**State 1**	**State 2**	**State 3**	**State 4**	**Est. Mean Reward**	**Stdev. Mean Reward**
State 1	100	100	0	0	500.0	0
State 2	100	100	0	0	500.0	0
State 3	100	100	0	0	500.0	0
State 4	100	100	0	0	500.0	0

3–4. Reward-based learning, i.e., learning with 10 levels improves performance, increased positive the effect of additional scenarios.
With additional systems, it is possible to exploit the problem with fewer scenarios more thoroughly.
With LHS the benefits of additional reward levels are more predictably apparent.
The effects are likely not significant.
8–10. Reward-based learning and additional systems both have positive effects. Of course, the random and nominal policies are not affected by features of the method including the number of systems and the reward-based learning levels.
10. Predictably, the optimal policy achieves the lowest objective and benefits greatly from added reward learning levels. The epsilon greedy methods are less able to benefit from additional reward levels since they cannot continually follow the optimal policy. Feature-based learning does benefit from additional levels. However, by ignoring transition information, its performance is generally highly suboptimal.

For the learning efficiency as evaluated by the MELT, findings from Fig. 5 include:

1. Adding scenarios increases the MELT values since the scenario probability is dispersed among additional options. With LHS, scenarios are differentiated resulting in reduced MELT values.
2. Additional subsystems learning together reduces the MELT since the systems pool their information.

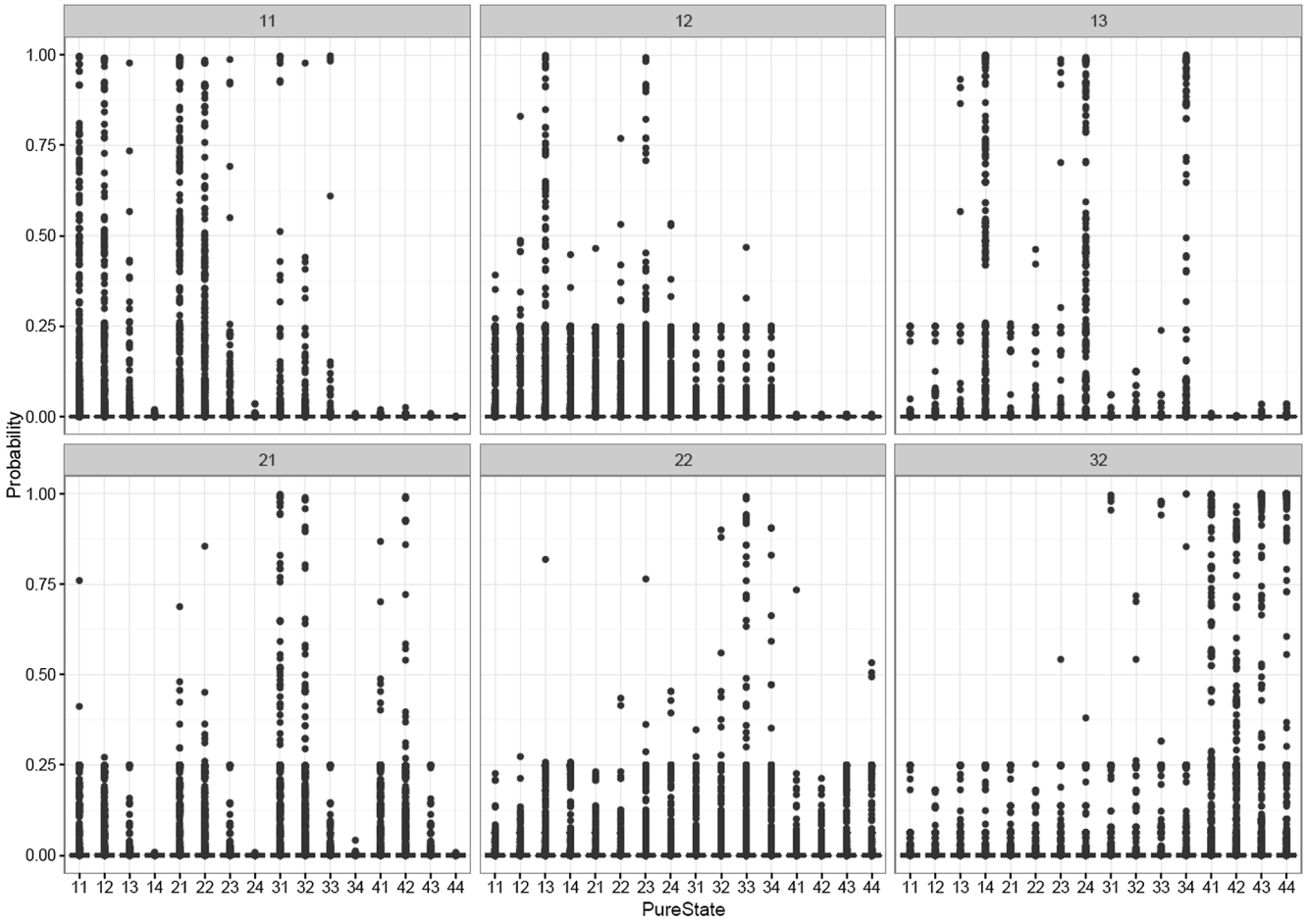

Fig. 6. Boxplots showing the belief point probabilities associated with each compound action by the putative optimal alpha vectors.

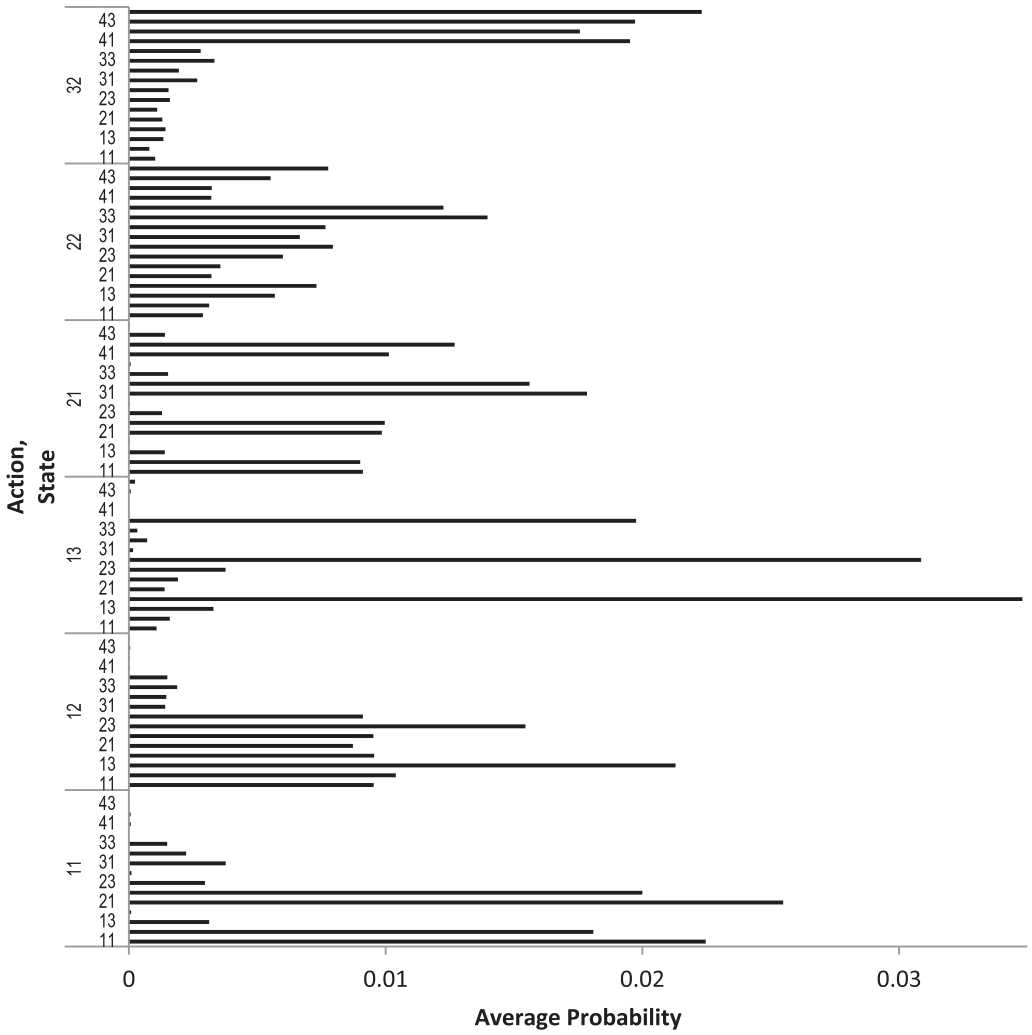

Fig. 7. Optimal actions and corresponding average probability of pure states in belief points.

3. Reward-based learning (# levels > 1) helps to significantly reduce the MELT.
4. Of course, the nominal policy never learns since it always uses Action #1. Its MELT is infinity. Random search learns best because it continues to experiment even after a sufficient amount is known to identify promising policies unlike the optimal policy.
5. Random scenarios decrease the MELT because scenarios can be like each other making full discrimination difficult. Having two systems sharing information reduces the effect.
6. Reward-based learning generally decreases the MELT, but random scenarios potential similarity decreases the effect.
7. Random sampling is presumably less affected by random scenarios than LHS because random sampling continues to learn over a longer period.
8. Reward-based learning positively interacts with subsystem information sharing to hasten learning and reduce the MELT.
9. Learning with two systems does less to reduce the MELT than might be expected.
10. Reward-based learning substantially reduces the MELT as would be expected. The random search learns the fastest (explores the most widely) followed by the epsilon greedy methods (which are partly random) and the optimal policy achieves a relatively high average MELT. This occurs presumably because the policy learns what it needs for objective value benefits and then stops learning if appropriate.

Overall, the LHS sampling makes the system behave similarly to having additional scenarios. Also, the method innovations introduced here to model-based Monte Carlo Bayesian reinforcement learning (reward-based learning and pooling subsystem information) significantly improve objective function performance and enhance learning (reducing the MELT values).

6. Case study: cyber vulnerability maintenance

In this section, we describe the application of the proposed methods (finite sampled scenarios, identical systems, and reward-based learning) to aid in decision-making about the maintenance of cyber vulnerabilities. We begin by describing the model data and assumptions. Then, we present the results and practical implications for decision-makers.

6.1. Model parameters for Linux hosts with critical data

Here, we focus on Linux hosts which have critical data inside the university firewall over an infinite horizon ($H = \infty$). We assume that hosts are in one of $N = 4$ states. Either they have (state 1) only low vulnerabilities, medium and possibly low vulnerabilities (state 2), high and possibly lower vulnerabilities (state 3), or critical and lower level vulnerabilities (state 4). These choices were determined by considering alternative logistic regression models to predict system incidents with

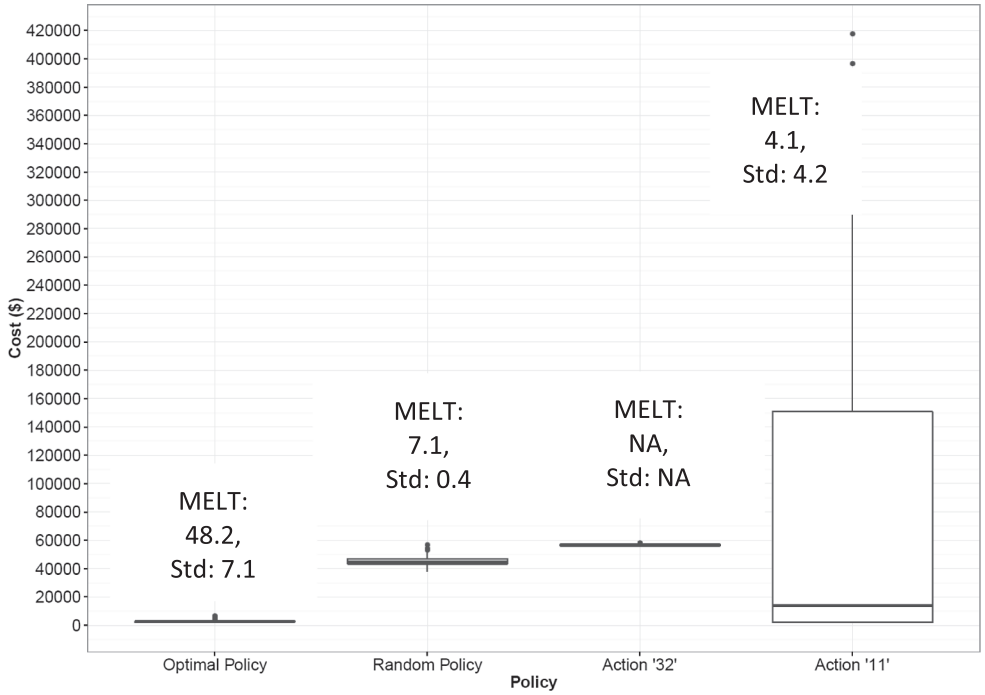

Fig. 8a. Comparison of simulation of average costs under different policies with 2 systems.

details omitted for brevity. N: Total number of "natural" system states (e.g., states of degradation).

Many organizations apply a simple policy to multiple types of hosts involving $u = 2$ action. They do not require any action (Action #1) for hosts in states 1–2 or send a staff member to investigate the system and attempt to apply a patch for states 3–4 (Action #2). We call Action #2 "research accept" because the organization often must accept the risk when research finds no patch available. Another action may or may not be part of the formal policy: Action #3 is essentially to lock down the hosts, shutting off access to the internet. We are referring to Action #3 as "compensating controls" that mitigate risks for critical hosts with known vulnerabilities. It is increasingly standard, perhaps, to apply compensating controls to critical vulnerabilities. Yet studying the implications for formal policy development can potentially reduce costs and aid in budgeting.

Here, we imagine that there are $W = 2$ approximately identical critical servers with critical data that are each managed individually by system administrators. The management approach greatly affects the costs in that organizations often greatly reduce costs by controlling host software from central images and not giving users administrative privileges. The tabulated transitions for twenty-one months of Linux hosts and estimated rewards are shown in Table 10. Note the success of limited efforts is the result of automatic patching from the vendors, making it less clear that local action is recommended. The expected costs are estimated using the rate of available repair personnel and predictions from a logistic regression model used to predict incidents with their accompanying average incident costs.

The underlines in Table 10 indicate that the available data are insufficient for an accurate estimate. A Latin hyper cube is generated with 10 runs for the 17 parameters creating the 10 scenarios, sampling from the Dirichlet distribution using Eq. (4). The compound transition matrices and expected costs are then created using Eqs. (5) and (6). Next, the right-hand-side of Table 3 is used to create the equivalent POMDP with observation matrices from Eq. (7) for $g = 10$ levels. The POMDP is then solved using PERSEUS and 10,000 belief points generated from 100 steps after 100 random starting points. The resulting alpha vectors predict the expected cost and recommended actions for all points in the belief space.

6.2. Results and implications

The alpha vectors generated by PERSEUS create a recommended set of actions and a budget. Each of the six alpha vectors in our putative optimal solutions includes 160 elements.

Fig. 6 depicts the probabilistic properties of the putative optimal solution derived by PEREUS. The bottom axis for each subplot includes all the possible pure states for the two subsystems. The solution involves 2000 belief points. The dots are the probabilities within a single belief state associated with a given action. The top left block shows the belief points under compound action 11. For all actions and all pure states, over 75% of the probabilities are near zero so we are studying the outliers. From the diagram it is apparent the action 11 is optimal for many belief states with relatively higher probabilities of the pure states 11, 12, 21 or 22. This is intuitive considering that action 1 is "limited effort" and states 1 and 2 are less severe in terms of vulnerabilities.

In Fig. 6, the block at the bottom right corner shows that compound action 32 is optimal for belief points with relatively higher probabilities in pure states 41, 42, 43 and 44. Here the optimal policy is (intuitively) suggesting the 'Compensating Control' action when there is critical vulnerability (state 4).

Given that so many of the belief state probabilities are near zero, studying the outliers primarily in Fig. 6, might be misleading. Therefore, we also include the average probabilities for each action and the most likely pure states in Fig. 7.

The lengths of the bars in Fig. 7 provide an approximate guide to which action should be taken in which situation. To compare the effectiveness of the putative optimal policy, we simulate the alpha vectors. Fig. 8a shows the average cost after running simulation on four different policies with two systems. In the simulation, we start from a specific belief point and take the optimal action at that point. The subsequent belief point is randomly picked based on the probability of states of the current belief point and the transition and observation probabilities following Eq. (11). The simulation is run for 1000 steps repeated 10 times. The "optimal policy" takes the optimal action at each belief point. The "random policy" takes actions randomly. The policy of "Action 32" chooses the action "32" at all belief points, and similarly "Action 11" chooses action 11. From the boxplot it is evident that the optimal policy is associated with the least average cost. The conservative

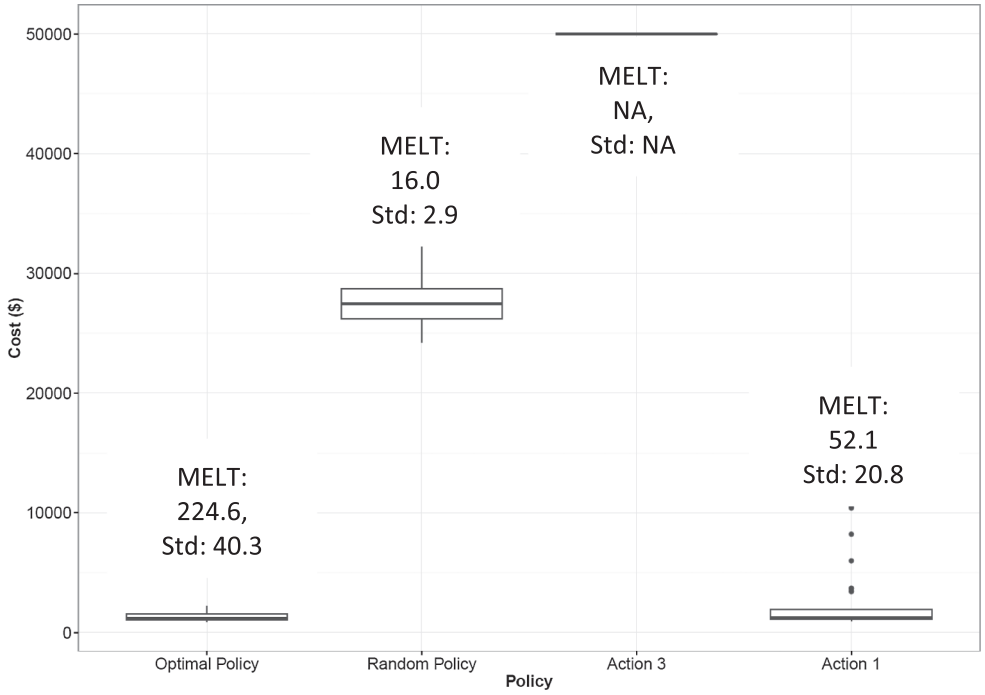

Fig. 8b. Comparison of simulation of average costs under different policies with single system.

policy of always choosing action "32", i.e. "Compensating Control" for system 1 and "Research Accept" for system 2 would be more expensive than choosing actions randomly. The average cost of the policy of always choosing action "11" would vary significantly based on the belief point the system is in. From Figs. 8a and 8b, it is evident that the optimal policy would incur the minimum expected cost by a margin among the other policies. The difference is many thousands of dollars per Linux host over its lifetime in maintenance and incident response costs.

The MELT values and standard deviation indicate the number of steps the simulation takes to converge to one scenario. The optimal policy converges with a MELT value of 48.2, whereas the random policy converges with a MELT value of 7.1. The "Action 32" policy does not converge at all (within 1000 simulation steps), and the "Action 11" policy converges with a MELT value of 4.1. Fig. 8b shows similar policies but for a single system setup. It is worth noting that the MELT for the optimal policy for single system is much higher (224.6) than that of the two systems. The same is true for all the other policies where MELT could be calculated. This is an example of the 'fast' learning aspect of multiple identical system setup.

7. Discussion

In this article, we consider a Bayesian reinforcement learning problem to address decision-making in data-limited environments motivated by cyber security maintenance policy design. We make five (relatively) major methodological contributions.

1. We introduce **variation reduction** techniques in scenario selection to BRL. With variation reduction techniques, fewer scenarios can more effectively approximate infinite scenario cases with provably reduced prediction variance, i.e., higher accuracy, leading to improvements in solver efficiency.
2. We explore a special type of multi-task reinforcement learning involving **identical systems**. For this type, we show the benefits of exact optimal reinforcement learning compared with methods which only learn from features, i.e., feature-based methods.
3. We introduce a method to measure the learning rate called the **Median Estimated Learning Time** (MELT). The MELT relates to the number of periods before parametric uncertainty is approximately eliminate or at least greatly reduced.
4. We show how model based learning can learn simultaneously from transitions, as others have done, and from **reward-based** learning. In our computational experiments, we show the benefits of reward-based learning in reduced expected discounted costs and faster learning times, i.e., reduced average MELT values.
5. We **prove** the intuitive relationship between model-based finite scenario reinforcement learning and a specific partially observable Markov decision process (POMDP) formulation. This rigorously establishes that which Duff (2002) and others assumed.

In our computational results, we compare three alternative systems (optimal, nominal-robust, and random) under sixteen conditions using a full factorial computational experiment. Relating to expected discounted costs (value), using identical systems and reward-based learning dominate. Also, the basic model-based Bayesian reinforcement learning (BRL) formulation is provided explicitly and not in recursive form. This permits the proof that the BRL formulation is equivalent to both a combined system BRL formulation for identical systems multitasking and a partially observable Markov decision (POMDP) process formulation. Therefore, an optimal solution to the associated POMDP is optimal for the BRL and identical system formulation. Also, steady state solutions are not only possible even while the system is learning, they are widespread and achievable at least to a tight approximation at 10,000 belief points.

The case study application to maintenance policies for critical Linux servers provided benefits both for budgeting and in clarifying that limited or no action (Action "11", "12", "21", "22") is appropriate for hosts with medium or lower vulnerabilities. For hosts with high or lower vulnerabilities, either research and accept or limited actions are appropriate, while for hosts with critical vulnerabilities immediate compensating control is appropriate. This differs from policies that have been applied in which patches are researched and risks are accepted if no patch is found.

There are multiple opportunities for future research. First, exact or approximate methods extending to large numbers of identical systems (again a form of multi-task methods) using normal approximations can

be explored. Additional applications within cyber security can be explored including control systems for firewalls or intrusion prevention and forensic systems to study and address intruders. Outside of cyber security, applications to improve the performance of medical alarm systems, scheduling, inventory design, and pricing can be considered. For many cases in which significant parametric uncertainty exists for relevant control factor combinations, e.g., price sensitivities, the proposed "fast" BRL methods can be applied and/or extended to generate desirable policies. Finally, more efficient methods to solve POMDPs than PERSEUS can be developed and/or applied to address larger numbers of systems states, actions, identical systems, and scenarios.

References

Alagoz, O., Ayvaci, M. U., & Linderoth, J. T. (2015). Optimally solving Markov decision processes with total expected discounted reward function: Linear programming revisited. *Computers & Industrial Engineering, 87*, 311–316.

Afful-Dadzie, A., & Allen, T. T. (2016). Control charting methods for autocorrelated cyber vulnerability data. *Quality Engineering, 28*(3), 313–328.

Afful-Dadzie, A., & Allen, T. T. (2014). Data-driven cyber-vulnerability maintenance policies. *Journal of Quality Technology, 46*(3), 234.

Allen, T. T. (2011). *Introduction to Discrete Event Simulation and Agent-based Modeling: Voting Systems. Health Care, Military, and Manufacturing.* Springer Publishing Company, Incorporated.

Allen, T. T., Sui, Z., & Parker, N. L. (2017). Timely decision analysis enabled by efficient social media modeling. *Decision Analysis, 14*(4), 250–260.

Ando, R. K., & Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research, 6*(Nov), 1817–1853.

Argyriou, A., Evgeniou, T., & Pontil, M. (2008). Convex multi-task feature learning. *Machine Learning, 73*(3), 243–272.

Bakker, B., & Heskes, T. (2003). Task clustering and gating for bayesian multitask learning. *Journal of Machine Learning Research, 4*(May), 83–99.

Bellman, R. E. (1957). *Dynamic Programming.* Princeton: Princeton University Press.

Calandriello, D., Lazaric, A., & Restelli, M. (2014). Sparse multi-task reinforcement learning. In Advances in Neural Information Processing Systems (pp. 819–827).

Cheng, G. Q., Zhou, B. H., & Li, L. (2017). Joint optimization of lot sizing and condition-based maintenance for multi-component production systems. *Computers & Industrial Engineering, 110*, 538–549.

Cockburn, E. (2009). Websites here, websites there, websites everywhere..., But are they secure? *The Quaestor Quarterly, 4*(3), 1–4.

Clarke, R. A., & Papadopoulos, E. (2016). Conclusion: Key Themes for the Next President. *The ANNALS of the American Academy of Political and Social Science, 668*(1), 212–217.

Chen, S. X., & Liu, J. S. (1997). Statistical applications of the Poisson-binomial and conditional Bernoulli distributions. *Statistica Sinica,* 875–892.

Delage, E., & Mannor, S. (2010). Percentile optimization for Markov decision processes with parameter uncertainty. *Operations Research, 58*(1), 203–213.

Duff, M. O. G. (2002). *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes (Doctoral dissertation.* University of Massachusetts Amherst).

Evgeniou, T., Micchelli, C. A., & Pontil, M. (2005). Learning multiple tasks with kernel methods. *Journal of Machine Learning Research, 6*(Apr), 615–637.

Ghavamzadeh, M., Mannor, S., Pineau, J., & Tamar, A. (2015). Bayesian reinforcement learning: A survey. Foundations and Trends®. *Machine Learning, 8*(5–6), 359–483.

Harrell, F. E., & Davis, C. E. (1982). A new distribution-free quantile estimator. *Biometrika, 69*(3), 635–640.

Hou, C. (2015). *Dynamic programming under parametric uncertainty with applications in cyber security and project management (Doctoral dissertation.* The Ohio State University.

Jafari, L., & Makis, V. (2016). Optimal lot-sizing and maintenance policy for a partially observable production system. *Computers & Industrial Engineering, 93*, 88–98.

Jalali, A., Sanghavi, S., Ruan, C., & Ravikumar, P. K. (2010). A dirty model for multi-task learning. In Advances in neural information processing systems (pp. 964–972).

Jawanpuria, P., & Nath, J. S., (2012). A convex feature learning formulation for latent task structure discovery In Proceedings of the 29th International Conference on Machine Learning, 2012

Kato, T., Kashima, H., Sugiyama, M., & Asai, K. (2010). Conic programming for multitask learning. *IEEE Transactions on Knowledge and Data Engineering, 22*(7), 957–968.

Lazaric, A., & Ghavamzadeh, M. (2010). Bayesian multi-task reinforcement learning. In ICML-27th International Conference on Machine Learning Omnipress (pp. 599–606).

Maurer, A., Pontil, M., & Romera-Paredes, B. (2013, February). Sparse coding for multitask and transfer learning. In International Conference on Machine Learning (pp. 343–351).

McKay, M. D., Beckman, R. J., & Conover, W. J. (1979). Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics, 21*(2), 239–245.

Misra, I., Shrivastava, A., Gupta, A., & Hebert, M. (2016). Cross-stitch networks for multi-task learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3994–4003).

Parisotto E., Ba J., & Salakhutdinov, R. (2016). Actor-mimic: Deep multitask and transfer reinforcement learning. In Proceedings of the 4th International Conference on Learning Representations.

Pineau, J., Gordon, G., & Thrun, S. (2003). Point-based value iteration: An anytime algorithm for POMDPs. *In IJCAI, 3*, 1025–1032.

Ponemon Institute (2017). cost of cybercrime study: United States. (2017). https://www.accenture.com/us-en/insight-cost-of-cybercrime-2017.

Poupart, P., Vlassis, N., Hoey, J., & Regan, K. (2006). An analytic solution to discrete Bayesian reinforcement learning. *Proceedings of the 23rd International Conference on Machine Learning,* 697–704.

Puterman, M. L. (2014). *Markov decision processes: Discrete stochastic dynamic programming.* John Wiley & Sons.

Ray, S., & Tadepalli, P. (2010). Model-based reinforcement learning. *Encyclopedia of Machine Learning,* 690–693.

Roychowdhury, S. (2017). Data-Driven Policies for Manufacturing Systems and Cyber Vulnerability Maintenance (Doctoral dissertation, The Ohio State University).

Ross, S., Pineau, J., Chaib-draa, B., & Kreitmann, P. (2011). A Bayesian approach for learning and planning in partially observable Markov decision processes. *Journal of Machine Learning Research, 12*(May), 1729–1770.

Shani, G., Brafman, R. I., & Shimony, S. E. (2006). Prioritizing point-based POMDP solvers. *European Conference on Machine Learning* (pp. 389–400). Berlin Heidelberg: Springer.

Smallwood, R. D., & Sondik, E. J. (1973). The optimal control of partially observable Markov processes over a finite horizon. *Operations Research, 21*(5), 1071–1088.

Spaan, M. T., & Vlassis, N. (2005). PERSEUS: Randomized point-based value iteration for POMDPs. *Journal of Artificial Intelligence Research, 24*, 195–220.

Srinivasan, R., & Parlikad, A. K. (2013). Value of condition monitoring in infrastructure maintenance. *Computers & Industrial Engineering, 66*(2), 233–241.

Sutton, R. S., & Barto, A. G. (1998). *Introduction to reinforcement learning, Vol. 135.* Cambridge: MIT Press.

Tang, B. (1993). Orthogonal array-based Latin hypercubes. *Journal of the American Statistical Association, 88*(424), 1392–1397.

Taylor, M. E., & Stone, P. (2009). Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research, 10*(Jul), 1633–1685.

Walraven, E. (2017), https://github.com/AlgTUDelft/SolvePOMDP (accessed 8-24-2018).

Wang, Y., Won, K. S., Hsu, D. and Lee, W. S. (2012). Monte Carlo Bayesian Reinforcement Learning. arXiv preprint arXiv:1206.6449.

Wiering, M., & Van Otterlo, M. (2012). Reinforcement learning. Adaptation, Learning, and Optimization, 12.

Wilson, A., Fern, A., Ray, S., & Tadepalli, P. (2007). Multi-task reinforcement learning: a hierarchical Bayesian approach. *Proceedings of the 24th international conference on Machine learning,* 1015–1022.

Zhang, Y., & Yang, Q. (2017). A survey on multi-task learning. arXiv preprint arXiv:1707.08114.