Technical Report

Remote Data Mirroring-A Three Objective Problem

June 24, 2021

Version: 0.9

1 Remote Data Mirroring

The self-adaptive Remote Data Mirroring (RDM) network [1] is based on the operational model presented in [2]. The RDM network under consideration consists of 25 RDM Mirrors (servers), to hold multiple copies of data, with 300 physical links in total that can used to transfer data between the mirrors. Each network link has an associated operational cost[1] and a measurable throughput, latency and loss rate used to determine the reliability, performance and cost of the RDM system [3, 4]. The goal here is to satisfy the non-functional requirements (NFRs) of Minimization of Costs (MC), Maximization of Performance (MP)[2] and Maximization of Reliability (MR) under environmental uncertainty of link failures and varying ranges of bandwidth consumption [1]. For this purpose, the network is required to continuously take adaptive actions of switching between the topological configurations of Minimum Spanning Tree (MST) and Redundant Topology (RT) to maintain better levels of satisfaction of NFRs. Both the configurations offer a different impact on the NFRs' satisfaction. The topological configuration of RT provides a higher level reliability than MST topology but it has a negative impact on the satisfaction of the MC and MP as the cost of maintaining non-stop RT topology will be high and due to data redundancy, the performance can be reduced. On the other hand, MST topology supports the satisfaction of MC and MP by maintaining a minimum spanning tree for the network.

This report is organized as follows. In Section 2, the techniques of Optimistic Linear Support (OLS) and Optimistic Linear Support with Alpha Reuse (OLSAR) along with Perseus MR-POMDP solver, DESPOT (a single objective POMDP solver) and DESPOT-ARROW are discussed. Section 3 presents the scenarios for experiments along with experiment results.

2 MR-POMDP Solvers

In this section, we give an overview of the multi-objective POMDP solver along with the techniques of OLS and OLSAR use to perform decision-making for the case of RDM network. The single objective techniques of DESPOT and DESPOT-ARROW are also discussed.

2.1 Persues POMDP Solver

Persues is a single objective POMDP solver based on the point-based Value Iteration framework (PBVI) as shown in Algorithm 1. In point based methods, the Value function V is represented in the form of α-vectors as follows:

$$V = \alpha = \begin{bmatrix} V(s_1) \\ V(s_2) \\ ... \\ V(s_n) \end{bmatrix} \tag{1}$$

and Value over belief is computed as:

$$V_b = \alpha.b \tag{2}$$

[1]In RDM system, Operational Cost is measured in terms of intersite network traffic. [3]

[2]In the case of RDM network, we are measuring performance in terms of total time to write the data i.e. the sum of the time to write each copy of data on each remote site. [3]

Algorithm 1: Perseus

```
Result: V
$B \leftarrow \emptyset$
$b \leftarrow b_0$
Repeat
select $a \in A$ randomly
$o = performAction(a)$
$b^{a,o} \leftarrow beliefUpdate(b, a, o)$
Add $b^{a,o}$ to B
$b \leftarrow b^{a,o}$
 Until $|B| \neq n$
Initialize $V \leftarrow V_0 \leftarrow \alpha_{min}$
Repeat $B' \leftarrow B$ $V' \leftarrow \emptyset$
Repeat
Randomly pick $b \in B'$
$\alpha_{new} \leftarrow PointBasedBackup(b, V)$
$if \alpha.b \geq V(b) then$
$B' \leftarrow b \in B' : \alpha.b < V(b)$
$\alpha_b \leftarrow \alpha_{new}$
else
$B' \leftarrow B' - b$
$\alpha_b \leftarrow argmax_{\alpha \in V} \alpha.b$
$V' \leftarrow V' \cup \alpha_b$
 Until $B' = \emptyset$
$V \leftarrow V'$
Until V converges
```

Perseus algorithm starts by creating a sample set of belief points by Random Exploration in belief space. During Random Exploration, an action is randomly selected and executed that results in an observation. For example, on the basis of the selected action a and observation o, a new belief sample point is computed using the belief update procedure as follows:

$$b^{a,o}(s') = Pr(s'|b, a, o) = \frac{Pr(s', b, a, o)}{Pr(b, a, o)} \tag{3}$$

$$= \frac{Pr(o|s', b, a)Pr(s'|b, a)Pr(b, a)}{Pr(o|b, a)Pr(b, a)} \tag{4}$$

$$= \frac{O(a, s', o) \sum_{s \in S} Pr(s'|b, a, s)Pr(s|b, a)}{Pr(o|b, a)} \tag{5}$$

where

$$Pr(o|b, a) = \sum_{s \in S} b(s) \sum_{s' \in S} T(s, a, s')O(a, s', o) \tag{6}$$

After creating the belief sample set, the initial value function represented in the form of α-vectors is computed as follows:

$$R_{min} = minR(s, a) \tag{7}$$

$$\alpha_{min} = R_{min}/1 - \gamma \tag{8}$$

$$V_0 = \{\alpha_{min}\} \tag{9}$$

Once the belief sample set and the initial set of α-vectors are computed, a point-based backup is performed iteratively to compute an optimal α-vector. During each iteration, a set B'=B and a new Value function V'=$\emptyset$ are created. One belief sample point is selected at random from B'and is used to compute a new α-vector by performing the point-based Bellman backup. For example, for k+1 iteration, the backup can be calculated as follows:

$$backup(b, V) = \arg\max_{\alpha^{b,a}_{k+1}} b.\alpha^{b,a}_{k+1} \tag{10}$$

where

$$\alpha_{k+1} = r^a + \gamma \sum_{o \in \Omega} \arg\max_{g^{a,o}} b.g^{a,o} \tag{11}$$

$$g_i^{a,o} = \sum_{s' \in S} O(a, s', o) T(s, a, s') \alpha_i(s') \tag{12}$$

where $g^{a,o}$ are the back-projections for all actions a and observations o of each next stage α-vector i.e $\alpha_i \in \alpha_k$.

If $\alpha_{new}.b > V(b)$, then α_{new} is added to V' and all other belief sample points $b' \in B'$ whose value is improved by α_{new} i.e. $\alpha_{new}.b' > V(b')$ are removed from B' so that they are not backed up at the current iteration. On the other hand, if $\alpha_{new}.b < V(b)$ then $\alpha_{old} = argmax_{\alpha \in V} \alpha.b$ is added to V'. The iteration ends when B'= $\emptyset$ and V is set to V'.

The process is repeated until V converges i.e. the difference between the V(b) from the current iteration and previous iteration is greater than the precision parameter η.

2.2 Multi-objective Perseus

In order to solve multi-objective decision-making problems, multi-objective reinforcement learning techniques are used having a vector-valued reward function **R**, instead of a scalar reward value. Hence, point-based methods represent the Value function is represented in the form of α-matrix **A** instead of α-vector to accommodate multiple objectives as follows:

$$\mathbf{A} = \begin{bmatrix} obj1 & obj2 & ... & objn \\ V(s1) & V(s1) & ... & V(s1) \\ V(s2) & V(s2) & ... & V(s2) \end{bmatrix} \tag{13}$$

Therefore, in Multi-objective Perseus algorithm, the point-based backup is re-defined to return an α-matrix. It takes the set of sampled belief points and initial set of α-matrix and performs the backup computation of the Value vector by computing the *vectorized* back-projections $\mathbf{G}^{a,o}$ for each next stage α-matrix $\mathbf{A}_i \in A_k$. The point-based backup for multi-objective perseus is computed as follows:

$$backup(b, \mathbf{V}) = \arg\max_{\mathbf{A}_{k+1}^{b,a}} b.\mathbf{A}_{k+1}^{b,a} \tag{14}$$

where

$$\mathbf{A}_{k+1} = r^a + \gamma \sum_{o \in \Omega} \arg\max_{\mathbf{G}^{a,o}} b.\mathbf{G}^{a,o} \tag{15}$$

$$\mathbf{G}_i^{a,o} = \sum_{s' \in S} O(a, s', o) T(s, a, s') \mathbf{A}_i(s') \tag{16}$$

2.3 Evaluation Parameter

In order to assess quality of approximation, point-based POMDP solvers use ϵ error that depends on the density δ_B of the belief sample set. δ_B represents the maximal distance from the closest $b \in B$ to other belief points in the belief sample set.

The ϵ error of the value of infinite-horizon POMDP after convergence of point-based methods is computed as follows:

$$\epsilon \leq \frac{\delta_B(R_{max} - R_{min})}{1 - \gamma^2} \tag{17}$$

where R_{max} and R_{min} are the maximum and minimum immediate rewards.

2.4 Optimistic Linear Support

The Optimistic Linear Support (OLS) algorithm is based on Cheng's Linear Support [5] approach for POMDPs. OLS presented as Algorithm 1 follows an outer loop approach that creates an outer shell around a POMDP solver (Persues) [6] to create a solution set known as the Convex Coverage Set (CCS) X [7] to represent the collection of value-vectors $\mathbf{V}^{\pi}$(specifying the multi-objective values) and their associated policies π such that after performing scalarization a maximizing policy is in the set. In order to select the policy having the maximizing value $V_X^*(w)$, linear scalarization of the value vector is performed using the parameters in the form of weights vector [3]. OLS helps in finding of these weights intelligently at runtime. The OLS algorithm considering a two objective problem has been presented in figure 1.

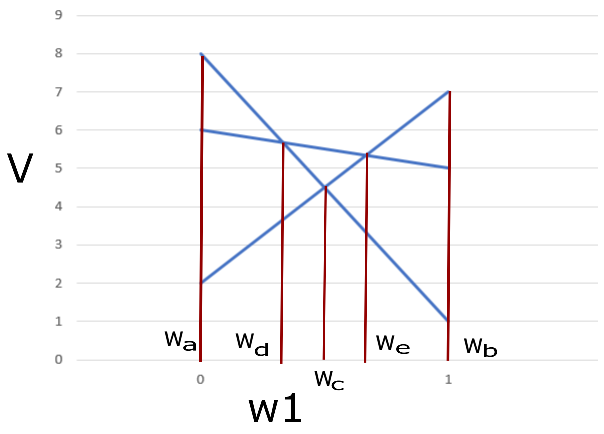


Figure 1: Optimistic Linear Support

The OLS algorithm starts by taking an empty set of X denoting the CCS of value-vectors as shown in line 1 of Algorithm 2. The algorithm repeatedly executes steps 2 to 9 until no improved value-vectors are found evaluated by the Maximal Possible Improvement Δ [8, 9]. In the first two iterations of the while loop, the algorithm selects the first two corner points as the extrema of the weights simplex i.e. w_a= 0.0 and w_b=1.0 as represented by the red vertical lines in figure 1. Next, the value-vectors, for example V_a= [8,7] and V_b=[2,7], for these corner points (w_a and w_b) are computed using a POMDP Solver represented by the blue lines in between these corner points. On the basis of these value vectors, a new corner point (w_c) is identified at their intersection. Then maximal possible improvement Δ [8] is computed for w_c. If Δ is improving then for this new corner weight w_c, a new value vector V_c= [6,5] is calculated by the calling the POMDP solver. More corner points are generated such as w_d and w_e from the intersecting points of the existing value vectors. Again Maximal Possible Improvement Δ for each w_d and w_e is calculated. The corner weight (out of w_d and w_e) having high Δ is selected and POMDP solver is called again to compute the value-vector for the selected weight. The process is repeated until none of the remaining corner weights yield an improvement in the form of Δ.

The process returns a set X known as CCS of the value vectors and their associated policies. As we have more than one policy returned, we have to select the best policy on the basis of scalarization function by taking weight values generated as part of the OLS algorithm. The policy having the maximum scalarized value $V_X^*(w)$ is then selected.

[3]Considering a two objective problem, the value of one of the weights can be computed as one minus the other weight due to the application of linear scalarization as w_1= 1- w_2 Where w_1 and w_2 refer to the scalarization weights for the values associated with objective 1 and objective 2 respectively.

Algorithm 2: Optimistic Linear Support

```
Result: Convex Coverage Set X
$X \leftarrow \emptyset$;
while $\neg(\Delta.isMaximum())$ do
    Select w ; // Select Corner Point w
    // Calculate Value-Vector V and $\pi$
    $(V, \pi) \leftarrow MRPOMDPSolver(w)$;
    $Calculate\Delta$;
    if $\Delta.isMaximum()$ then
        $X \leftarrow X \cup V$;
    end
end
```

The flow chart representing the step by step execution of OLS algorithm along with Multi-objective Perseus Algorithm is shown in figure 2.

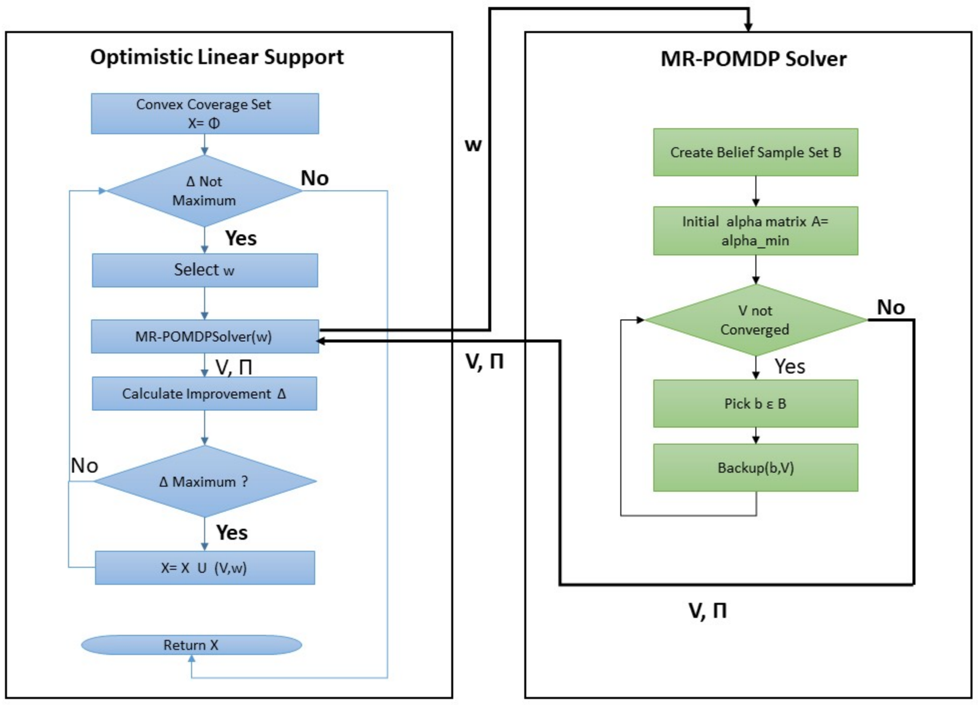


Figure 2: Optimistic Linear Support with Multi-Reward POMDP Solver

2.5 Optimistic Linear Support with Alpha Reuse (OLSAR)

The OLSAR algorithm is an extension of OLS algorithm. It follows the same steps as OLS but it reuses the alpha matrices from previous iterations to compute the approximate Convex Coverage Set (CCS) of value vectors.

3 DESPOT Solver

Determinized Sparse Partially Observable Tree (DESPOT) [10, 11] is an algorithm for online POMDP planning under uncertainty. Next, relevant details of the algorithm are presented.

1. **Build a DESPOT tree to project future evolutions of NFRs**.

The algorithm considers future evolutions of the state of a system to decide the next action $a \in A$, i.e. to reason about long-term effects of immediate actions [11]. The future evolutions of the state of the system are represented by the following DESPOT tree.

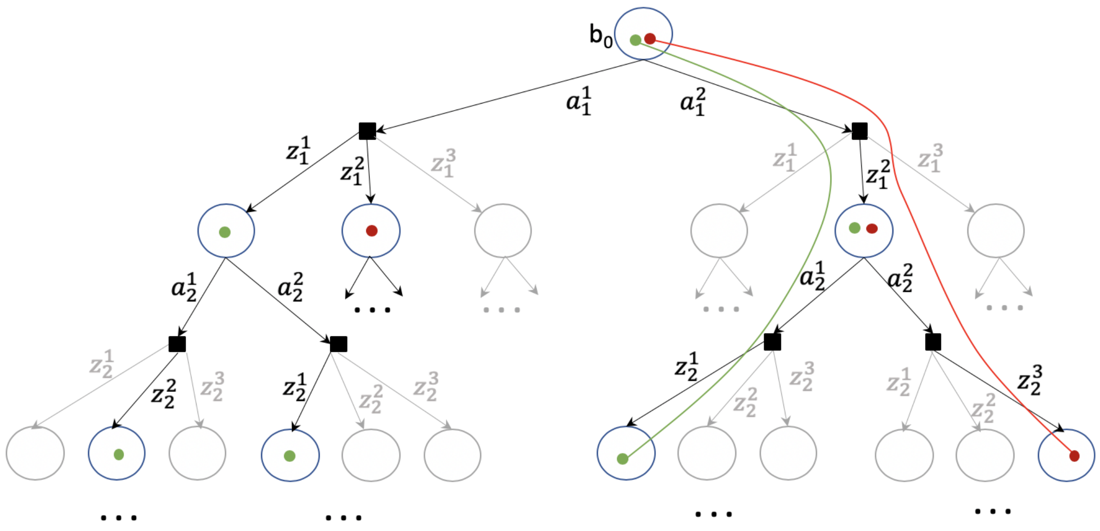


Figure 3: DESPOT Belief Tree with 2 sampled scenarios marked with green and red dots (The DESPOT tree is overlaid on a standard belief tree)

DESPOT builds per each time slice, a sparse approximation of a standard belief tree: a DESPOT tree (See Fig. 3), by using a simulation model [11]. The root node of the tree is the belief b_0 which represents the belief about the current state of the running system. Each edge in the tree represents an action observation pair. If a child node b_t is connected to its parent b_{t-1} by an edge (a_t, z_t), then b_t is computed by using Bayesian inference [11]. The DESPOT tree represents the neighborhood of the current belief b_0.

2. **Select an optimal action** $a \in A$.

   The Bellman's principle of optimality is shown in equation (18). It is applied over a DESPOT tree to choose the best action:

$$V(b) = \max_{a \in A} \left\{ \sum_{s \in S} b(s)R(s,a) + \gamma \sum_{z \in Z} p(z|b,a)V^*(r(b,a,z)) \right\} \tag{18}$$

   The algorithm searches the tree with root at the current belief b_0. DESPOT uses lookahead search [11] to approximate the optimal discounted reward value $V^*(b_0)$ [10]. The search is guided by a lower bound $l(b_0)$ and an upper bound $\mu(b_0)$ on the approximated optimal discounted reward value $V^*(b_0)$. The explorations continue, until the gap between the bounds $\mu(b_0)$ and $l(b_0)$ reaches a target level ϵ_0 or the allocated planning time T_{max} runs out. Equation (18) recursively computes over the tree the maximum value of action branches and the average value of observation branches [12, 10]. The result is an approximately optimal policy for the current belief b_0 [10, 11]. The system then executes the first action of the policy $\pi(b_0)$, i.e. the action with the highest discounted reward value $V^*(b_0)$.

The algorithm 3, which is shown below, provides a high-level view of the process to build and search a DESPOT tree. Two additional algorithms (Algorithms 4 and 5), complement this process by implementing specific behaviours for exploration and backup of belief nodes b, to determine an optimal action, which is derived from the DESPOT tree.

```
Algorithm 3: Algorithm for building and searching a DESPOT tree D in each time slice
  Parameter(s):
  - Initial belief $b_0$ about the current state of the system
  Runtime execution:
  -Use the belief $b_0$ to create the root node of a new DESPOT tree D
  -Initialize upper and lower bounds: $\mu(b_0)$ and $l(b_0)$
  -Initialize $\epsilon(b_0) \leftarrow \mu(b_0)$ - $l(b_0)$
  while $\epsilon(b_0) > \epsilon_0$ and running time is less than $T_{max}$ do
  b $\leftarrow$ Explore(D,$b_0$) (Explore a promising path)
  Backup(D,b) (Backup on bounds at each node b)
  Return: DESPOT tree with an approximated optimal policy $\pi^*(b_0)$
```

In morel detail, the algorithm 3 constructs and searches a DESPOT tree incrementally. Initially, it contains only a root node with belief b_0 about the current state of a system. The tree also contains the initial upper and lower bounds associated to the belief b_0. The algorithm performs explorations using algorithm 4, to expand the DESPOT tree and to reduce the gap $\epsilon(b_0)$ between the bounds $\mu(b_0)$ and $l(b_0)$ at the root node b_0. Each exploration aims at choosing and expanding a promising leaf node (line 8) and adds its child nodes into the tree until the current leaf node is not heuristically promising [11]. Then, the algorithm traces the path back to the root and performs backup using Algorithm 5 on the upper and lower bounds at each node along the way to the root node. The explorations continue until the gap between the bounds $\mu(b_0)$ and $l(b_0)$ reaches a target level ϵ_0 ($\epsilon_0 >= 0$) or the planning time T_{max} finishes.

```
Algorithm 4: Algorithm to expand the branches of a DESPOT tree D
  Parameter(s):
  -A DESPOT tree D and the current belief b Runtime execution:
  -$D_h \leftarrow$ DESPOT tree height
  -$\Delta$(b) $\leftarrow$ current height of the belief b while $\Delta(b) <= D_h$ and $E(b) > 0$ do
  if b is a leaf node in D then
  Expand b one level deeper. Insert each new child b' of b into D,
  and initialize $\mu(b')$ and $l(b')$
  $a^* \leftarrow \arg\max_{a \in A} \mu(b,a)$
  $z^* \leftarrow \arg\max_{z \in Z_{b,a^*}} E(\tau(b,a^*,z))$
  b $\leftarrow \tau(b,a^*,z^*)$
  Return: An expanded DESPOT tree
```

In Algorithm 4, the exploration to expand the DESPOT tree starts at the root node b_0. At each node b along the exploration path, the best action branch a^*, according to the upper bound $\mu(b)$, is selected. Afterwards, the observation branch z that leads to a child node $b' = \tau(b, a^*, z)$ maximizing the excess uncertainty $E(b')$, is selected. The excess uncertainty $E(b')$ measures the difference between the current gap at b' and the expected gap at b' if the target gap $\epsilon(b_0)$ at b_0 is satisfied. The exploration strategy seeks to reduce the excess uncertainty in a greedy manner [11]. The exploration at a node b is terminated under the following conditions. First, Δ(b) $<= D_h$, i.e. the maximum tree height is exceeded, and second, $E(b) > 0$, indicating that when the expected gap at b is reached, further exploration from b onwards may be unproductive. When the exploration terminates, Algorithm 5 is performed.

Algorithm 5: Algorithm to perform backup on the bounds of each node b using Bellman's principle

```
Parameter(s):
-A DESPOT tree D and the current belief b
Runtime execution:
for each node x on the path from b to the root D do
    Perform backup on $\mu(x)$ and l(x)
```

In Algorithm 5, the path back to the root node is traced, and the backup is performed on the upper and lower bounds at each node b along the way, using the Bellman's principle of optimality [11]. Specifically, the bounds $\mu(b)$ and $l(b)$ at each node b are recomputed.

4 Experiments

Experiments for the case study were performed using both OLS and OLSAR algorithms along with multi-objective Perseus Solver for 100 time steps.

4.1 Initial Setup

The components of the MR-POMDP for the considered RDM network are explained as follows:

States: As we represent states as combinations of satisfaction levels of NFRs, for the three NFRs (MEC, MR and MP), eight states are identified and are shown in Table 1.

Actions: represent the adaptation strategies to maintain the satisfaction of the NFRs of MC, MR and MP. The actions for the case of RDM network are the two topological configurations of Minimum Spanning Tree (MST) and Redundant Topology (RT).

Rewards: There is a vector-valued reward function to represent the priorities of NFRs from the perspective of the experts. Therefore, the reward vector has a size of 3 represented as **R(s,a)**=[R$_{MC}$,R$_{MR}$,R$_{MP}$]. R$_{MC}$ represents the priority value for MC, R$_{MR}$ represents the priority values for MR and R$_{MP}$ represents the priority values for MP at runtime. The reward vector values (provided by the experts) for NFRs in the RDM network are shown in Table 2

Transition Function: According to Rule 1, the states are represented as a combination of NFRs in MR-POMDP. The transition probabilities T(s,a,s') are factored as marginal conditional probabilities of NFRs $P(MC' \mid MC, a)$, $P(MR' \mid RPL, a)$ and $P(MP' \mid MP, a)$ using the property of conditional independence and Bayes rule [13] as follows:

$$T(s, a, s') = P(s'|s, a) = P(MC' \mid MC, a)P(MR' \mid MR, a)P(MP' \mid MP, a)$$

The transition probabilities for going from one state to another as a result of action for the RDM case are shown in the Tables 3, 4 and 5. These transition probabilities are provided by the experts.

Observations: As the states of NFRs are not directly observable, we use monitorables to obtain observations required for monitoring the satisfaction levels of NFRs on the basis of the set of possible values or information obtained from the environment. For this purpose, three monitorable variables Ranges of Bandwidth Consumption (RBC), Active Network Links (ANL) and Total Time for Writing (TTW) related to the NFRs MC, MR and MP respectively are specified in the RDM system. Higher the value for the ANL, higher will be satisfaction of MR. On the other hand, lower the values of RBC and TTW, higher will be the satisfaction of MC and MP. In RDM System, all of these monitorable variables have range boundaries that are specified in the requirements specifications by the experts as shown in Table 6.

In MR-POMDP, the observations of each monitorable are described by the probability distributions over its possible values. Hence, Observation probability represented by O(s',a,z)=P(z —a,s') species the probability of observing the observation z after executing action a and transition to state s'.

Like the transition model, we also factor the observation model into the product of conditional probabilities [13] computed as:

$$O(s', a, z) = P(z|s', a) = P(Mon_1, ..Mon_n|s', a)$$

Hence,

$$P(z|s', a) = P(RBC, ANL, TTW|s', a) = P(REC|s', a)P(ANL|s', a)P(TTW|s', a)$$

The conditional probabilities for the observation model are shown in Table 6.

Table 1: States of Priority-Aware MR-POMDP for RDM Network

S	**NFR$_1$=MC**	**NFR$_2$=MR**	**NFR$_2$=MP**
s_1	True	True	True
s_2	True	True	False
s_3	True	False	True
s_4	True	False	False
s_5	False	True	True
s_6	False	True	False
s_7	False	False	True
s_8	False	False	False

Table 2: Reward Values for the NFRs in the RDM Network

S	**Action(A)**	**Reward Vector Values**		
		R$_{NFR1} = R_{MC}$	R$_{NFR2} = R_{MR}$	R$_{NFR3} = R_{MP}$
s_1	MST	39.17	39.0	40.0
s_2	MST	41.0	40.0	39.0
s_3	MST	39.0	38.0	38.5
s_4	MST	17.0	16.0	15.0
s_5	MST	44.0	43.0	43.5
s_6	MST	29.0	28.0	27.0
s_7	MST	14.0	13.0	13.5
s_8	MST	2.0	1.0	1.0
s_1	RT	41.0	43.0	41.0
s_2	RT	32.0	33.0	31.0
s_3	RT	28.0	29.0	27.0
s_4	RT	26.0	27.0	25.0
s_5	RT	28.0	29.0	27.0
s_6	RT	16.0	17.0	15.0
s_7	RT	23.0	24.0	22.0
s_8	RT	11.0	12.0	10.0

Table 3: Transition Probabilities for NFR MC

NFR$_1$=Minimization of Cost (MC)					
Action(A)	**MC$_t$**	**MR$_t$**	**MP$_t$**	**P(MC$_{t+1} = T$)**	**P(MC$_{t+1} = F$)**
MST	True	True	True	0.9	0.1
MST	True	True	False	0.88	0.12
MST	True	False	True	0.92	0.08
MST	True	False	False	0.9	0.1
MST	False	True	True	0.85	0.15
MST	False	True	False	0.83	0.17
MST	False	False	True	0.87	0.13
MST	False	False	False	0.85	0.15
RT	True	True	True	0.86	0.14
RT	True	True	False	0.84	0.16
RT	True	False	True	0.88	0.12
RT	True	False	False	0.86	0.14
RT	False	True	True	0.73	0.27
RT	False	True	False	0.71	0.29
RT	False	False	True	0.75	0.25
RT	False	False	False	0.73	0.27

Table 4: Transition Probabilities for NFR MR

NFR$_2$=Maximization of Reliability (MR)					
Action(A)	**MC$_t$**	**MR$_t$**	**MP$_t$**	**P(MR$_{t+1} = T$)**	**P(MR$_{t+1} = F$)**
MST	True	True	True	0.91	0.09
MST	True	True	False	0.93	0.07
MST	True	False	True	0.89	0.11
MST	True	False	False	0.91	0.09
MST	False	True	True	0.93	0.07
MST	False	True	False	0.95	0.05
MST	False	False	True	0.91	0.09
MST	False	False	False	0.93	0.07
RT	True	True	True	0.95	0.05
RT	True	True	False	0.97	0.03
RT	True	False	True	0.93	0.07
RT	True	False	False	0.95	0.05
RT	False	True	True	0.97	0.03
RT	False	True	False	0.99	0.01
RT	False	False	True	0.95	0.05
RT	False	False	False	0.97	0.03

Table 5: Transition Probabilities for NFR MP

$\mathbf{NFR_1}$=Maximization of Performance (MP)					
Action(A)	**$\mathbf{MC}_t$**	**$\mathbf{MR}_t$**	**$\mathbf{MP}_t$**	**$\mathbf{P(MP}_{t+1} = T)$**	**$\mathbf{P(MP}_{t+1} = F)$**
MST	True	True	True	0.9	0.1
MST	True	True	False	0.85	0.15
MST	True	False	True	0.92	0.08
MST	True	False	False	0.87	0.13
MST	False	True	True	0.88	0.12
MST	False	True	False	0.83	0.17
MST	False	False	True	0.9	0.1
MST	False	False	False	0.85	0.15
RT	True	True	True	0.82	0.18
RT	True	True	False	0.75	0.25
RT	True	False	True	0.84	0.16
RT	True	False	False	0.77	0.23
RT	False	True	True	0.8	0.2
RT	False	True	False	0.73	0.27
RT	False	False	True	0.82	0.18
RT	False	False	False	0.75	0.25

Table 6: CPT RBC:$P(RBC \mid MC, a)$, ANLs:$P(ANLs \mid MR, a)$ and TTW:$P(TTW \mid MP, a)$

$\mathbf{Mon_1}$=Ranges of Bandwidth Consumption(RBC)				
Action(A)	**$\mathbf{MEC}_t$**	**$\mathbf{P(RBC}_{t+1} < x)$**	**$\mathbf{P(MC}_{t+1} in[x, y])$**	**$\mathbf{P(MC}_{t+1} >= y)$**
MST	True	0.8	0.15	0.05
MST	False	0.72	0.18	0.1
RT	True	0.78	0.16	0.06
RT	False	0.68	0.2	0.12
$\mathbf{Mon_2}$=Active Network Links(ANLs)				
Action(A)	**$\mathbf{MR}_t$**	**$\mathbf{P(ANLs}_{t+1} < r)$**	**$\mathbf{P(ANLs}_{t+1} in[r, s])$**	**$\mathbf{P(ANLs}_{t+1} >= s)$**
MST	True	0.06	0.16	0.78
MST	False	0.12	0.2	0.68
RT	True	0.05	0.15	0.8
RT	False	0.1	0.18	0.72
$\mathbf{Mon_3}$=Time to Write(TTW)				
Action(A)	**$\mathbf{MR}_t$**	**$\mathbf{P(TTW}_{t+1} < f)$**	**$\mathbf{P(TTW}_{t+1} in[f, g])$**	**$\mathbf{P(TTW}_{t+1} >= g)$**
MST	True	0.83	0.13	0.04
MST	False	0.67	0.23	0.1
RT	True	0.8	0.15	0.05
RT	False	0.63	0.25	0.12

4.2 Parameter Setting

As both OLS and OLSAR use point based planning Persues Solver, the quality of approximation is evaluated using ϵ error. As infinite horizon MR-POMDPs are undecidable, it is difficult to compute the true CCS, we have compared the results of the algorithms with reference sets generated with large number of belief sample points and used it to select the belief sample set size for the test runs as shown in Fig. 4 and 5 respectively.

For OLS, the MR-POMDP model converges with the belief sample set size of 100 with ϵ value of 0.01002345 as compared to the test runs executed with 300, 500, 700 and 1000 samples respectively. Similarly for OLSAR algorithm, MR-POMDP model using the belief sample set of size 100 converges first having the minimal ϵ error of 0.01281355 as compared to the test runs executed with 300, 500, 700 and 1000.

4.3 Experiment Scenarios

Experiments were performed by introducing different scenarios of the environmental conditions under which the RDM network works as follows:

Stable Scenario: representing the RDM network performing under normal environmental conditions

Detrimental Scenario representing the RDM network performing under the dynamic environment when different disturbance levels (increased packet loss or network link failures) in the network environment are introduced. There are two cases of detrimental situations that have been studied:

Case 1 Introduction of disturbance levels when MST is the selected topology:

Random changes (disturbance levels) in the environment during the execution of the MST topology are introduced to decrease the reliability of the system $P(MR_{t+1} = True|NFR_t, MST_t)$

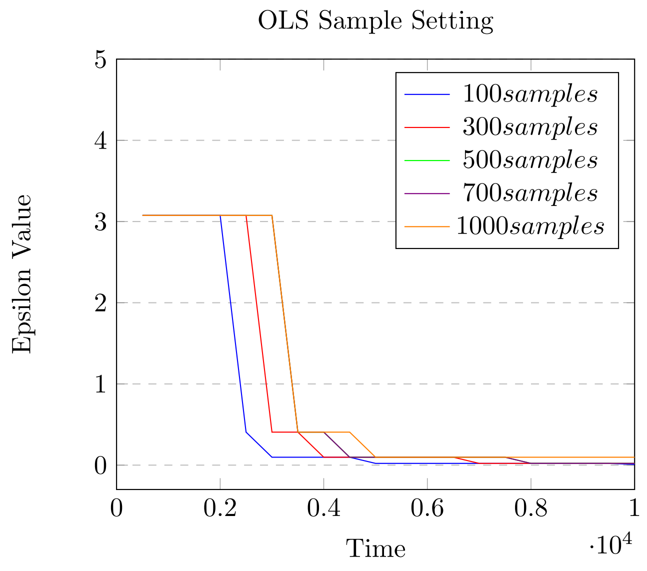


Figure 4: Parameter Setting using Epsilon Value

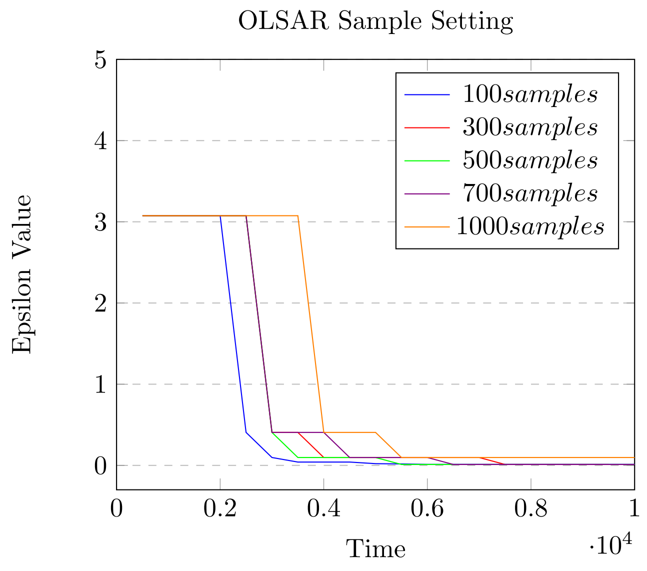


Figure 5: Parameter Setting using Epsilon Value

Description: An unexpected data packet loss during the execution of MST generates a decrease in the reliability of the system. A MST topology connects all remote sites in the RDM network by the identification of a minimum spanning tree on the network of links among each remote site. Data packet loss may represent network link failures in the RDM system, which may be caused due to problems with the equipment such as failure of a switch or a router.

Case 2 Introduction of disturbance levels when RT is the selected topology.

Random Changes (disturbance levels) in the environment during the execution of the RT topology are introduced to increase the cost and to decrease the performance of the system: $P(MC_{t+1} = True, MP_{t+1} = True | NFR_t, RT_t)$.

Description: Unexpected data packet loss during the execution of the RT Topology, are generating an unusual rate of data forwarding, which would increase cost in terms of bandwidth consumption and would decrease the system's performance. In case of RDM , the cost for inter-site links communication refers to the data sent over them. Therefore, during RT topology, that involves a bigger number of inter-site network links than a MST, is more expensive. Costs increase as the number of network links increases and a decrease in the system's performance could also be expected.

Next, the experiments for both OLS and OLSAR using stable and detrimental scenarios are presented. The experiments were performed for 100 timesteps.

4.4 OLS Algorithm: Experimental Evaluations

In this section, we present the results of experiments performed using OLS algorithm under both the initially specified stable scenario and the dynamic context Case 1 (DC1) and dynamic context Case 2 (DC2). We also present, a comparison of the results with the approaches of DESPOT and ARROW.

4.4.1 Stable Scenario

In stable scenario, both OLS and DESPOT show comparable results by keeping the satisfaction of NFRs MC, MR and MP above the required satisfaction threshold as shown in Fig.6. Both the approaches, show a preference for MST topology than RT topology under stable conditions as shown in Fig.7.

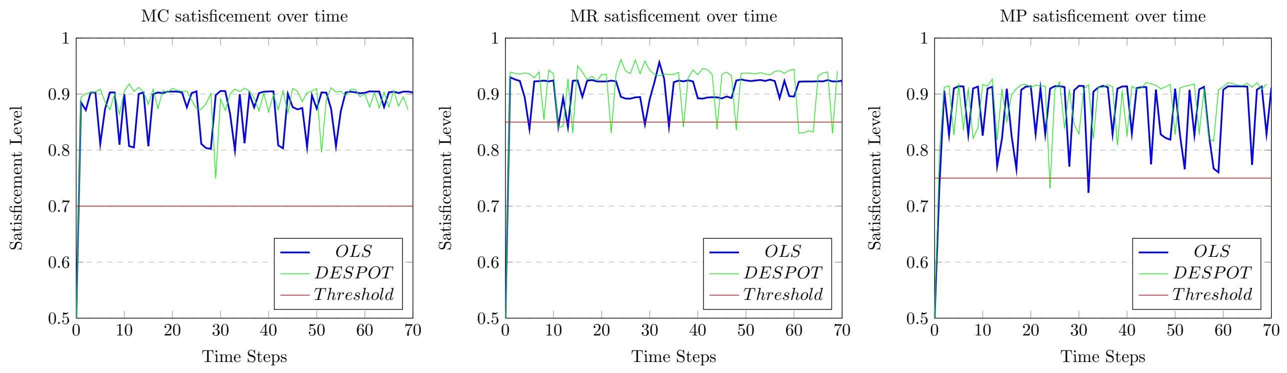
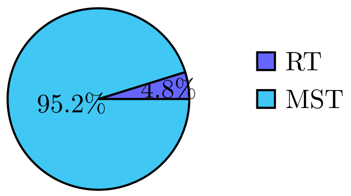


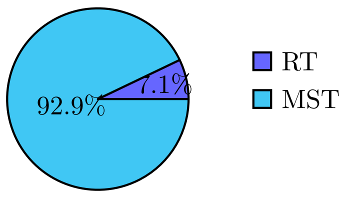
Figure 6: Satisfaction of NFRs over time under Stable Scenario

4.4.2 Dynamic Context 1

Dynamic Context 1 (DC1) represents the changes in the environment during the execution of MST Topology that has an impact on the reliability of the system. For this purpose, we have simulated deviations in the transitions of MST topology to study the impact on the reliability of the system. During DC1, OLS shows



(a)



(b)

Figure 7: Topology Selection using a) OLS and b) DESPOT

MC in good zone of satisfaction by having the satisfaction levels above the threshold value throughout the timeline whereas for the satisfaction level of MR and MP goes below the threshold a several time steps as shown in Fig.8. Although, the satisfaction level of MP in case of OLS, is going below the threshold value but it remains closer to the boundary of satisfaction. On the other hand, despite of showing good satisfaction for MC and MP, DESPOT shows poor satisfaction level of MR throughout the timeline as compared to OLS. For the purpose of further evaluation, we have also compared our results with the technique of ARROW that supports DESPOT with the update of reward values at runtime. However, ARROW shows good satisfaction level for MR, it shows poor satisfaction for MP at several time steps. Both the techniques of OLS and DESPOT show an increase in the usage of RT topology to 30.0 and 21.4 percent respectively under the dynamic context DC1 as compared to the RDM network performing under stable context as shown in Fig.9. On the otherhand, for ARROW the preferred topology is RT to offer support to the satisfaction of MR.

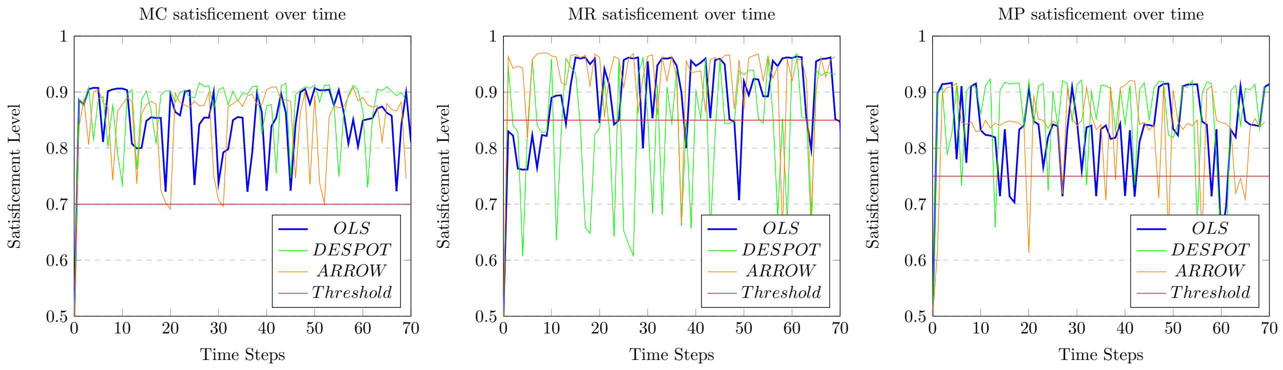
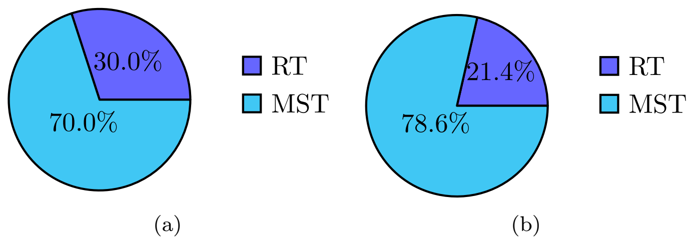


Figure 8: Satisfaction of NFRs over time under Dynamic Context 1

4.4.3 Dynamic Context 2

Dynamic Context 2 (DC2) represents the changes in the environment during the execution of RT Topology that has an impact on the cost and performance of the system. For this purpose, we have simulated deviations in the transitions of RT topology to study the impact on the bandwidth consumption (cost) and performance of the system. During DC2, OLS shows the best results in terms of satisfaction of all of the NFRs MC, MR and MP by keeping them in their suitable zone of satisfaction throughout the timeline as shown in Fig.10. In order to achieve the required satisfaction levels, OLS increases the usage of MST topology to 99.4 percent under the dynamic context DC2 as compared to stable scenario as shown in Fig.11. DESPOT also shows comparable resultsIn contrast, ARROW doesn't show good satisfaction results for MC and MP as their satisfaction levels go below the threshold value at several points of time. The preferred topology for both DESPOT and ARROW under DC2 remains MST having the selection percentage of 92.9 percent for DESPOT and 68.6 percent for ARROW.



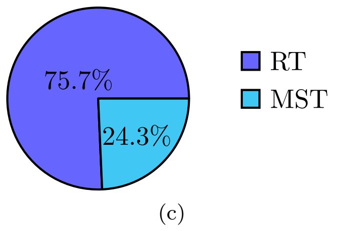


Figure 9: Topology Selection using a) OLS and b) DESPOT c) ARROW

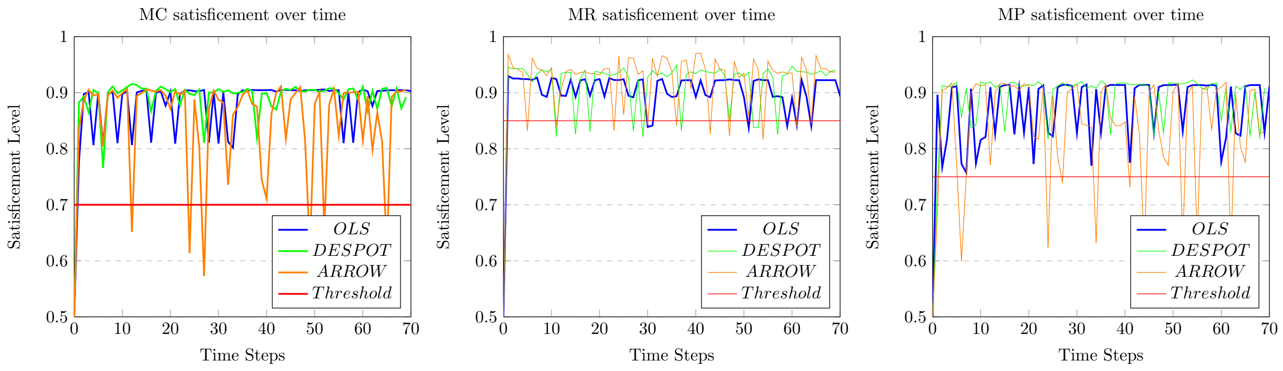


Figure 10: Satisfaction of NFRs over time under Dynamic Context 2

4.5 OLSAR Algorithm: Experimental Evaluations

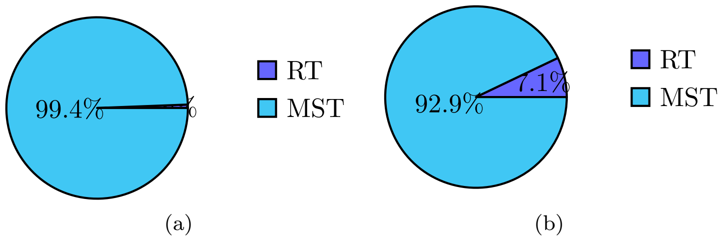
In this section, we present the results of experiments performed using OLSAR algorithm under both stable and the dynamic contexts of DC1 and DC2. We also present, a comparison of the results with the approaches of DESPOT and ARROW.

4.5.1 Stable Scenario

In stable scenario, both OLSAR and DESPOT show comparable results by keeping the satisfaction of NFRs MC, MR and MP above the required satisfaction threshold as shown in Fig.12. Both the approaches, show a preference for MST topology than RT topology under stable conditions as shown in Fig. 13.

4.5.2 Dynamic Context 1

During DC1, OLSAR shows MC and MP in good zone of satisfaction by having the satisfaction levels above the threshold value whereas for the satisfaction level of MR goes below the threshold at several time steps as shown in Fig. 14. On the other hand, despite of showing good satisfaction for MC and MP, DESPOT shows poor satisfaction level for MR throughout the timeline as compared to OLSAR. On the other hand, the technique of ARROW shows good satisfaction level for MR but in order to support MR, it shows poor



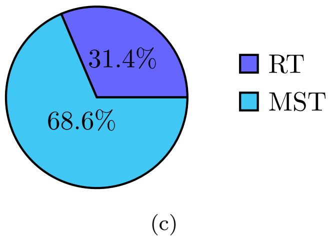


Figure 11: Topology Selection using a) OLS and b) DESPOT c) ARROW

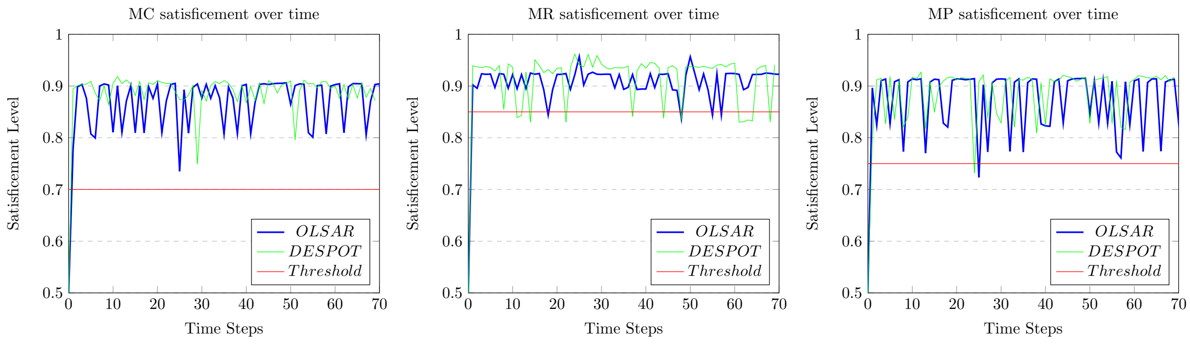


Figure 12: Satisfaction of NFRs over time under Stable Scenario

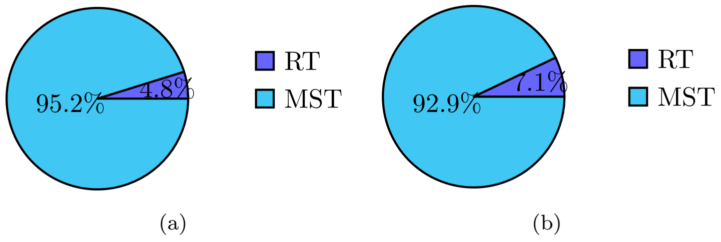


Figure 13: Topology Selection using a) OLSAR and b) DESPOT

satisfaction for MP at several time steps. Both the techniques of OLSAR and DESPOT show an increase in the usage of RT topology to 35.8 and 21.4 percent respectively under the dynamic context *DC1* as compared to the RDM network performing under stable context as shown in Fig.15. In contrast, ARROW shows a preference of RT topology over MST topology to offer support for the satisfaction of MR.

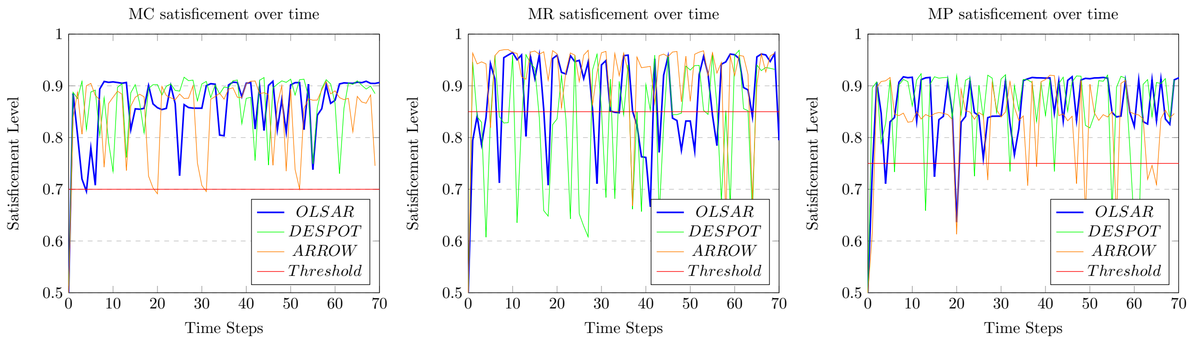
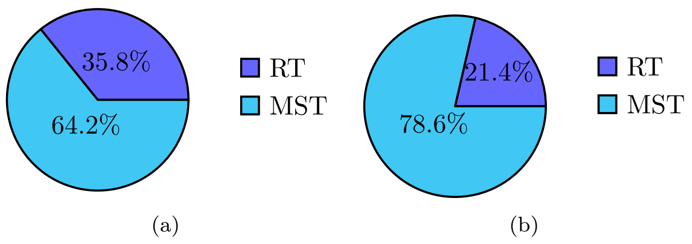


Figure 14: Satisfaction of NFRs under Dynamic Context 1



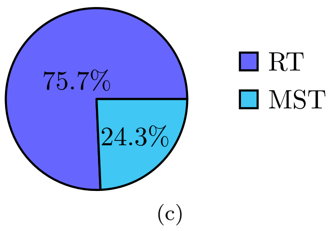


Figure 15: Topology Selection using a) OLSAR and b) DESPOT c) ARROW

4.5.3 Dynamic Context 2

During DC2, OLSAR shows the best results in terms of satisfaction of all of the NFRs MC, MR and MP by keeping their satisfaction level above the threshold throughout the timeline as shown in Fig.16. However, the satisfaction of MR goes below the threshold value at a few points of time but it remains closer to the threshold. For this purpose, OLSAR increases the usage of MST topology to 99.8 percent under the dynamic context DC2 as shown in Fig.17. DESPOT also shows comparable results In contrast, ARROW doesn't show good satisfaction results for MC and MP as their satisfaction levels go below the threshold at several timesteps. The preferred topology for both DESPOT and ARROW under DC2 remains MST having the selection percentage of 92.9 percent for DESPOT and 68.6 percent for ARROW.

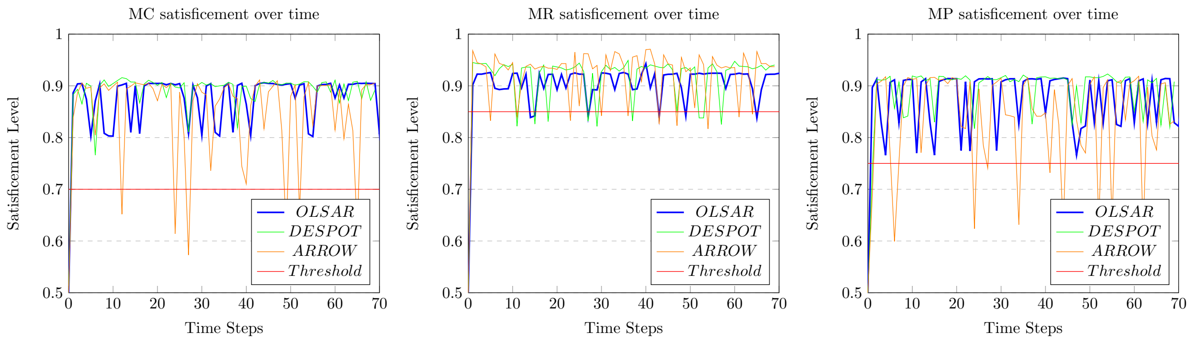
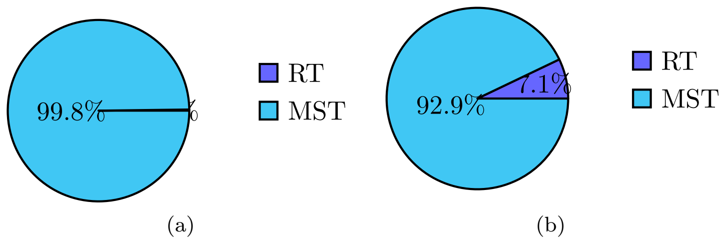


Figure 16: Satisfaction of NFRs over time under Dynamic Context 2



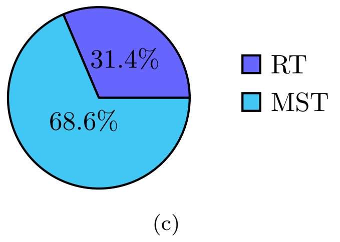


Figure 17: Topology Selection using a) OLSAR and b) DESPOT c)ARROW

5 Summary

To sum up, the overall average satisfaction of all the NFRs, by both the algorithms of OLS and OLSAR, under both the stable and dynamic contexts DC1 and DC2 is above the satisfaction threshold as shown in Fig.18 and 19.

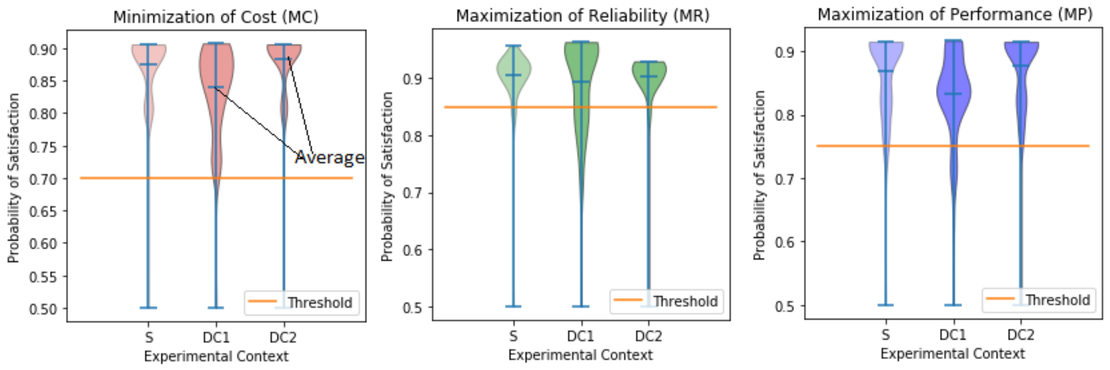


Figure 18: Average Satisfaction of Non-Functional Requirements using OLS under Stable (S), Dynamic Context 1 (DC1) and Dynamic Context 2 (DC2)

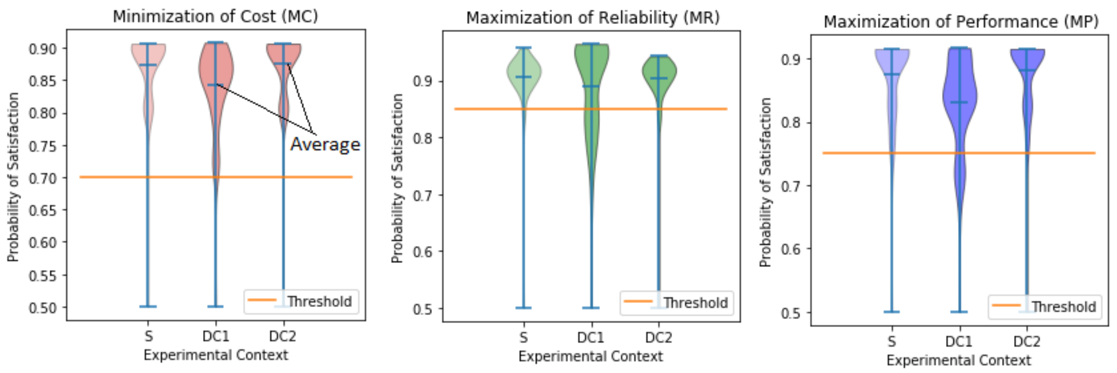


Figure 19: Average Satisfaction of Non-Functional Requirements using OLSAR under Stable (S), Dynamic Context 1 (DC1) and Dynamic Context 2 (DC2)

References

[1] M. Ji, A. C. Veitch, and J. Wilkes, "Seneca: remote mirroring done write," in *USENIX Annual Technical Conference, General Track*, 2003.

[2] K. Keeton, C. Santos, D. Beyer, J. Chase, and J. Wilkes, "Designing for Disasters," Mar. 2004.

[3] L. Garcia-Paucar and N. Bencomo, "Knowledge base k models to support trade-offs for self-adaptation using markov processes," *13th IEEE Conference on Self-Adaptive and Self-Organizing Systems. Umea, Sweden*, 2019.

[4] N. Bencomo and L. Garcia-Paucar, "Ram: Causally-connected requirements-aware models using bayesian inference," *IEEE/ACM 22nd International Conference on Model Driven Engineering Languages and Systems (MODELS). Munich, Germany.*, 2019.

[5] H.-T. Cheng, "Algorithms for partially observable markov decision processes," Ph.D. dissertation, University of British Columbia, 1988.

[6] M. T. J. Spaan and N. Vlassis, "Perseus: Randomized Point-based Value Iteration for POMDPs," *Journal of Artificial Intelligence Research*, vol. 24, pp. 195–220, Aug. 2005, arXiv: 1109.2145. [Online]. Available: http://arxiv.org/abs/1109.2145

[7] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A Survey of Multi-Objective Sequential Decision-Making," *Journal of Artificial Intelligence Research*, vol. 48, pp. 67–113, Oct. 2013. [Online]. Available: https://jair.org/index.php/jair/article/view/10836

[8] D. M. Roijers, S. Whiteson, and F. A. Oliehoek, "Point-Based Planning for Multi-Objective POMDPs," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, Jun. 2015. [Online]. Available: https://www.aaai.org/ocs/index.php/IJCAI/IJCAI15/paper/view/10939

[9] D. Roijers, *Multi-objective decision-theoretic planning*, 2016, oCLC: 6893481195.

[10] A. Somani, N. Ye, D. Hsu, and W. Lee, "Despot: online pomdp planning with regularization," *In Advances in Neural Information Processing Systems (NIPS)*, 2013.

[11] N. Ye, A. Somani, D. Hsu, and W. S. Lee, "Despot: Online pomdp planning with regularization," *Journal of Artificial Intelligence Research 58 (2017) 231-266*, 2017.

[12] S. Ross, J. Pineau, S. Paquet, and B. Chaib-Draa, "Online planning algorithms for pomdps," *J. Artificial Intelligence Research, 32(1), 663–704*, 2008.

[13] L. H. G. Paucar and N. Bencomo, "RE-STORM: mapping the decision-making problem and non-functional requirements trade-off to partially observable markov decision processes," in *Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems - SEAMS '18*. Gothenburg, Sweden: ACM Press, 2018, pp. 19–25. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3194133.3195537