Defence Research and Development Canada / Recherche et développement pour la défense Canada

CAN UNCLASSIFIED

DRDC | RDDC technologysciencetechnologie

Development of a Joint Intelligence and Information S&T Capability Task Authorization 22—Deployable Intelligence Source Collection Value Optimizer (DISCOVER)—Multi-Satellite Collection Scheduling

Closed-Loop Collection Tasking (Multiple Episodes)—Implementation

R. Tharmarasa
M. Florea
T. Kirubarajan
Thales Research & Technology (TRT) Canada

Prepared by:
Thales Research & Technology (TRT) Canada
1405, boul. du Parc Technologique
Québec, QC G1P 4P5

Contractor Document Number: 2066C.022-REP-04-ClosedL Rev. 01
PSPC Contract Number: W7701-125076/001/QCL
Technical Authority: Jean Berger, Defence Scientist, DRDC – Valcartier Research Centre

Contractor's date of publication: March 2018

Defence Research and Development Canada

Contract Report
DRDC-RDDC-2018-C189
October 2018

CAN UNCLASSIFIED

Canada

THALES

Development of a Joint Intelligence and Information S&T Capability
Task Authorization 22 – Deployable Intelligence Source Collection Value Optimizer (DISCOVER) – Multi-Satellite Collection Scheduling Closed-Loop Collection Tasking (Multiple Episodes) - Implementation

Contract No.: W7701-125076/001/QCL

Document Control No.: 2066C.022-REP-04-ClosedL Rev. 01

Date: 31 March 2018

THALES

Development of a Joint Intelligence and Information S&T Capability
Task Authorization 22 – Deployable Intelligence Source Collection Value Optimizer (DISCOVER) – Multi-Satellite Collection Scheduling
Closed-Loop Collection Tasking (Multiple Episodes) - Implementation

Contract No.: W7701-125076/001/QCL

Prepared by:
R. Tharmarasa (Thamas)
T. Kirubarajan (Kiruba)
Sensemaking Specialists

Approved by:
Steeve Côté
Project Manager, Thales Canada Inc.

REVISION HISTORY

Date	Rev.	Description	Author
31 March 2018	01	Final release	R. Tharmarasa Mihai Florea Thia Kirubarajan

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

D

DISCOVER	Deployable Intelligence Source Collection Value Optimizer

M

MDP	Markov Decision Process

P

POMDP	Partially Observable Markov Decision Process

T

TLE	Two-Line Element

This report is prepared by Thales Canada as a delivery for the Subtask 6.4 – “Closed-loop collection tasking (multiple episodes)” of the Task Authorization 22 – “Deployable Intelligence Source Collection Value Optimizer (DISCOVER) – multi-satellite collection scheduling” from the contract W7701-125076/001/QCL.

1 ABSTRACT

Closed-loop collection tasking does explicitly account for information feedback (e.g., observation outcome) interleaving task planning and task plan execution. Environment assumptions are typically characterized by:

- uncertainty,
- dynamic (on-line) conditional (real-time) task planning (sequential decision),
- multiple tasking episodes, and,
- a high tempo of operations (state transition rate /events) requiring responsiveness.

Dynamic tasking may be appropriate when operating under uncertainty, latency conditions or high operations tempo/state transition rate (e.g., time-dependent communication and decision cycle, observation outcome/feedback, resource availability/failure, urgent task occurrence, delayed observation and information feedback, information-sharing, weather conditions).

Through this document, we explore adaptive and contingency planning/scheduling concepts to the dynamic multi-satellite collection scheduling problem. We have also developed a new adaptive scheduling decision problem model properly dealing with state transitions (e.g., exogenous event, new task, feedback information, resource availability/failure, urgent observation requests) while planning and executing tasks and trading-off run-time and solution quality.

2 RÉSUMÉ

La tâche de collecte en boucle fermée tient en compte de façon explicite du retour d'information (e.g. le résultat de l'observation) entrelaçant la planification des tâches avec l'exécution du plan. Les hypothèses de l'environnement sont caractérisées habituellement par:

- incertitude,
- planification des tâches (décision séquentielle), conditionnelle (temps-réel) et dynamique (enligne)
- épisodes multiples d'assignation de tâches, et,
- une fréquence élevée d'opérations (transition d'états) qui nécessite une réactivité accrue.

L'assignation de tâches en mode dynamique peut être peut s'avérer très utile lorsqu'on opère sous incertitude, en conditions nécessitant des grands délais ou des grands temps de réponses, et lorsque la fréquence des d'opérations est élevée (e.g., communications et cycle décisionnel qui dépendent du temps, retour des observations ou du résultat, disponibilité ou défaut des ressources, rajout des tâches urgentes, retour d'observations et d'information en retard, partage d'informations, conditions météo).

À travers ce document, nous explorons des nouveaux concepts de planification adaptive et d'urgence pour le problème de planification dynamique de collections multi-satellites. Nous avons aussi developpé un nouveau modèle pour le problème de décision pour une planification adaptative qui traite correctement avec les transitions d'états (e.g., événements exogènes, nouvelles tâches, retour d'information, disponibilité/défaut des ressources, requêtes urgentes d'observation) lors de la planification et l'exécution des tâches, en trouvant un équilibre entre le temps de calcul et la qualité de la solution.

3 CONCEPT DEVELOPMENT

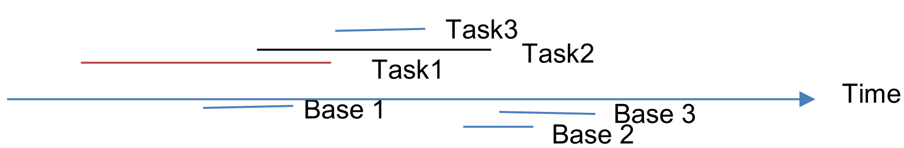
The multi-satellite imagery collection scheduling decision problem can be defined through the tuple *<R, C, Sppt, Obj, A, M, CONS >*, stated as follows:

- *R*: Set of collection tasks (requests);
- *C*: Set of collection assets (collectors);
- *Sppt*: Set of supporting resources – typically, ground stations;
- *Obj*: Collection tasking objective(s);
- *A*: Action space/set;
- *M:* Collection-assets-to-collection-tasks matching; and
- *CONS*: Constraints set.

Parameters:

H: Time horizon
X_s: state of earth observing satellite s
S: number of satellites
R_r: request r
Scan an AOI
Track a target
Identify/classify a target

For a given satellite orbit, possible actions and communication over time may be as follows.



Possible Task-satellite-time assignment can be calculated based on satellite orbit and the task location.

In order to handle any possible new tasks, we may need to calculate the possible coverage regions over time.

Since the communication between a satellite and a ground station is limited due to coverage and other constraints, we need to consider the constraints on download and upload.

3.1 Literature review

3.1.1 Exact and inexact scheduling algorithms for multiple earth observation satellites under uncertainties of clouds

- Tasks:
  - A target, i.e., a circle with limited dimension; and
  - A polygon which may cover a wide geographical area.
- Due to its large size, a polygon usually is failed to be observed in a single orbit and therefore **partitioned into multiple strips**;
- **Tasks are corresponding to the strips** that require being observed;
- If the scheduling horizon is long enough, a satellite will orbit the earth for multiple orbits and pass over a strip for multiple times;
- To handle the difficulties, we formulate the **orbits of the satellites as the resources**;
- Memory and the energy capacities for each orbit are formulated as constants:
  - Assume that there will be an opportunity for data downloading for each orbit, and the data will be all downloaded resulting in the memory becoming empty again;
- Then, the problem is to assign strips (task) to orbits:
  - T – set of tasks
  - O – set of orbits
  - P_i – profit of task i
  - M_k – memory capacity of orbit k
  - E_k- energy capacity of orbit k
  - m_k- memory consumption for each unit observation
  - e_k- energy consumption for each unit observation
  - b_{ik} – 1: if task i can be observed on orbit k, 0: otherwise
  - $[w_{sik}, w_{eik}]$ - time window for task i on orbit k
    - Start and end time are fixed. So, no need to find the start and end time:
      - Non-agile satellites.
    - Memory consumption for task i on orbit k is $(w_{eik} - w_{sik})m_k$
    - Energy consumption for task i on orbit k is $(w_{eik} - w_{sik})e_k$
  - θ_{ik}- slewing angle for task i
  - st_{ij}^k- setup time between task i and task j on orbit k
    - After observing a task, the satellite requires a sequence of transformation operations to observe the next one, which is sensor shutdown → slewing → attitude stability → startup.
  - ρ_{ij}^k- energy consumption for slewing from task i to task j
  - p_{ik}- probability that the task i will be successfully observed on orbit k
- The unknown variable is x_{ij}^k, assigning task i and j to orbit k, and task i is the immediate predecessor of task j.

Non-agile satellites

They have the maneuverability of rolling (slewing), but not pitching:

- Rolling (slewing): a movement that is perpendicular to the direction of the orbit; and
- Pitching: a movement along the direction of the orbit.

Agile satellites

They have the maneuverability of rolling (slewing) and pitching.

4 A PROBLEM FORMULATION FOR NON-AGILE SATELLITES

Assumptions:

- Requests are portioned into to multiple strips, which are called tasks;
- A task must be successfully scanned at least by one satellite:
    - In the open-loop planning, multiple satellites might be assigned to one task to improve the probability of above. But, in the closed-loop planning, we can assign the second orbit only if the first orbit is failed.

For non-agile satellites, start and end time for a task are fixed for a given orbit.

Ground station computes an optimal joint policy and communicates the individual policies or series of actions to perform to the corresponding satellites.

4.1 Do we need local policies for each satellite?

Unless a task can be stopped in the middle, the future action will not depend on the outcome of the last task until feedback is available from the ground station. Hence, ground station (global decision maker) can send the series of action to perform until the next communication.

If a task can be stopped in the middle, then time steps must be introduced instead of using the connected graph.

Another possibility for local policy is to handle environmental conditions. But we should be able to predict the environmental effect on a task based on local satellite measurements. If an environmental condition affects all the possible actions during a certain duration, then again policy is not useful. The policy is useful only if the predicted environmental condition affects only a part of the observation area.

Hence, we consider three cases:

1. No sensor/process is available on a satellite to measure the environmental effects:
    - Ground station just sends the actions to perform before the next communication.
2. A satellite can predict if a task can be completed successfully before it starts the task. The prediction may vary over time-based on additional measurement:
    - Hence, a local MDP is needed to change the tasks based on measurements of the environment or any other factor that affect performing a task.
3. A satellite can measure if the current task is successful so far. If it is not successful, the satellite can stop that task and start a new task:
    - Again, we need a local MDP to handle this case. But it is a harder problem than above case.

4.2 Global decision maker without local MDPs

The object is to assign tasks to orbits under energy and memory constraints.

4.2.1 Approximation 1: Orbit level policies

Instead of changing the series of actions after each upload/download, only the actions for the future orbits are changed.

State:

- Orbit number o
- State of task {0,1}

Actions:

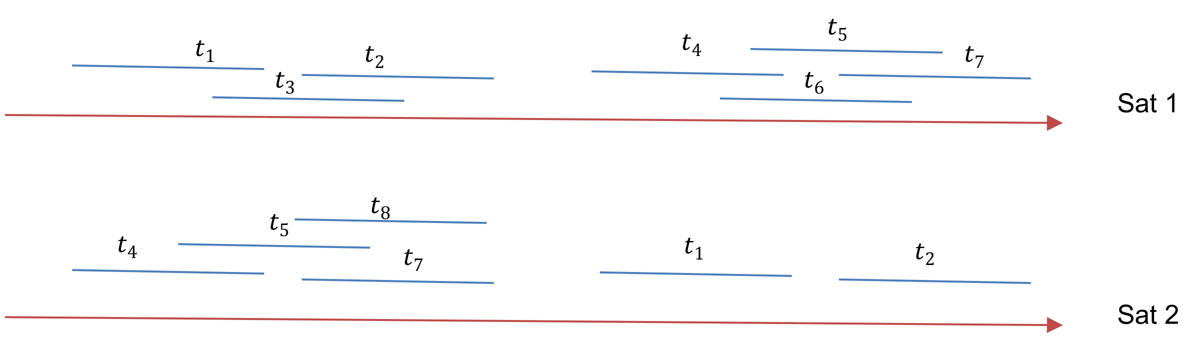
$\{a_{soi}\}$ $\forall i \& s$ for given orbit number o

$a_{soi} \in \{0,1\}$, $a_{soi} = 1$ means that task i must be performed.

The number of states is $2^{N_t} * N_o$, where N_t – number of tasks, N_o- number orbits. If $N^t = 100$ and $N_o = 3,$ the number of state becomes $3 * 2^{100}$.

4.2.1.1 Clustering

In order to reduce the state and action space, the tasks can be clustered, if they are sparse. For example, even if we have 100 tasks, if 10 tasks are for Ottawa region, 8 tasks for Toronto region, …, then tasks can be clustered.



In the above example, we can form two clusters {1, 2, 3} and {4, 5, 6, 7, 8}.

Now, we can define a sub-MDP for each cluster. The state and actions will be similar to above, but only the actions in the clusters are considered. If we have x clusters for 100 tasks with the maximum number of tasks in a cluster is 10, the worst-case state size is $10 * 2^{10}$.

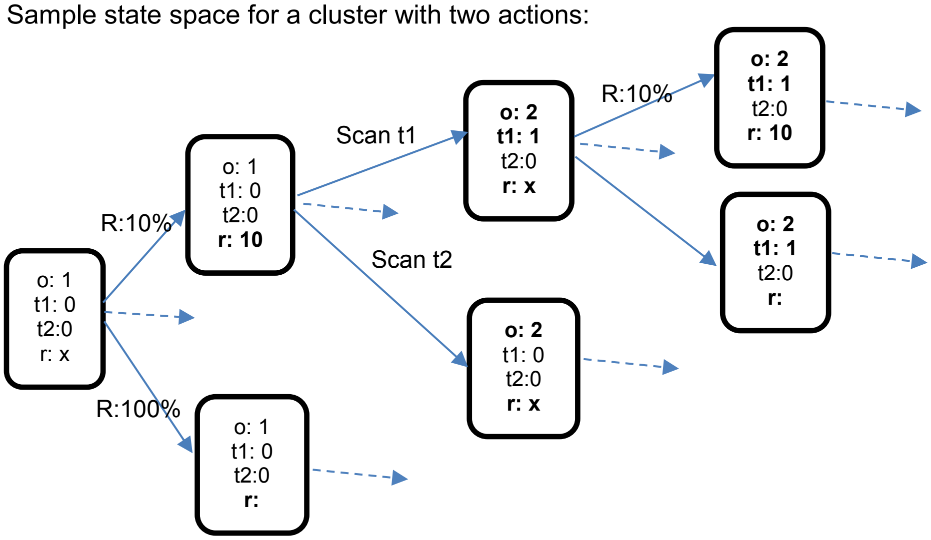
Even though tasks are disjoined, satellite resources are shared by clusters. Hence, somehow, we have to consider the assignment jointly. In order to joint sub-MDPs, we can add the remaining/allocated resources to the states of each sub-MDP.

If resources are allocated to each cluster, then each MDP can be solved independently.

Then, the state will contain:

- Orbit number o
- State of task {0,1}
- Allocated resources {10%, 20%, …, 100%}

Sample state space for a cluster with two actions:



Resource Allocation

If we assume that memory and energy consumption depends only on the time spent on tasks, rather than the task type, we can define a variable for percentage of resources, instead of percentage of energy and memory separately.

We need to allocate resources for each orbit and cluster.

First, we can find the total scan time for all the tasks in the given satellite. Then, allocated resource for orbit o for cluster i can be given as

$$R_i^o = discretize\ (T_i^o / \sum(T_j^o))$$

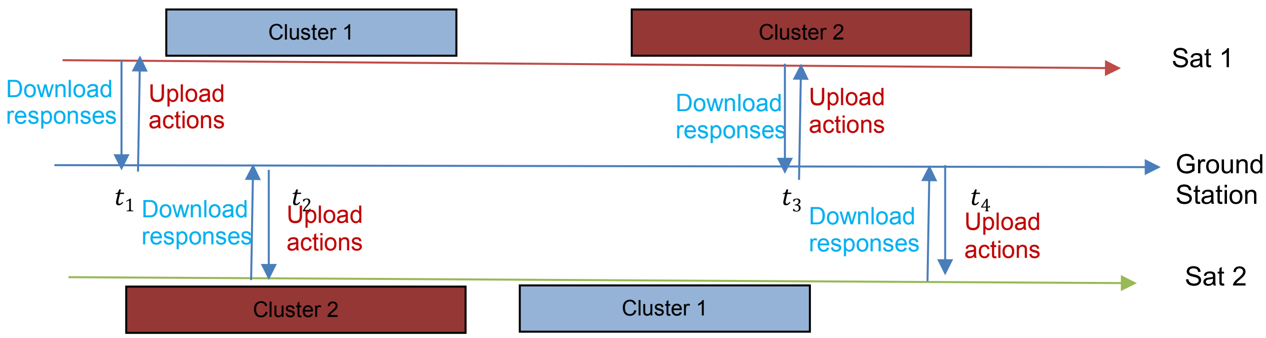
We may need to find a better way.

Also, in order to find the optimal solution to sub-MDPs, we need to know the transition probability $p(R_i^o, a, R_i^{o-1})$. In general, this transition probability depends on the result of other clusters. But, we may need to ignore this to simply the problem.

4.2.2 Approximation 2: Download level policies

Again, we can use the clustering proposed in the previous section.

Here is a possible scenario:



We need to define states at each possible download/upload times. Also, we may need to change the resource allocation after each download.

Challenges:

- For cluster 1, we get the response from Sat 1 at t_3, but Sat 2 is still processing, or the response is not known yet; and
- For cluster 2, only a partial response is available at t_3 or t_2.

1.1.1.1 State space

State:

- Download number d
  - Orbit number o
  - Satellite number s
- State of task {0,0.5,1}
  - 0:need to scan
  - 0.5: a satellite is processing
  - 1: done
- Satellite resource level {0, 0.1, .., 10}
  - Satellite failure can be handled by setting this to zero.

Actions:

$\{a_{sdi}\}\ \forall i$ for the satellite s downloaded now till the next download time

$a_{sdi} \in \{0,1\}$, $a_{sdi} = 1$ means that task i must be performed.

4.3 Global decision maker with local MDPs

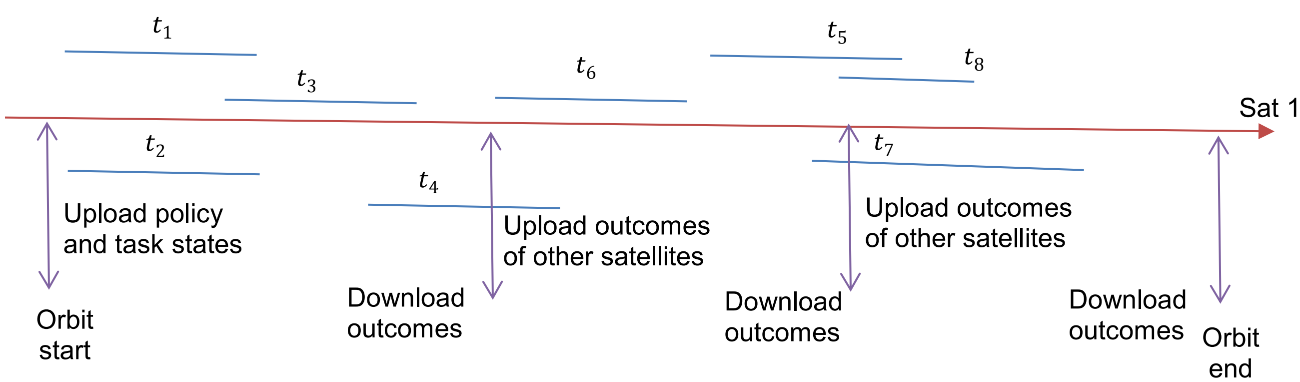
Global MDP finds policies for each satellite using the weather or other environmental forecasts available at the time of planning. However, the weather forecast changes over time, and it affects the transition probabilities. Instead of rerunning the global MDP, a local MDP can be used to improve the policies of the inter-communication intervals.

4.3.1 Formulation 1: Global MDP finds Policies for Each satellite

In this formulation, the global MDP finds the policies for each satellite for the entire horizon and sends the policy π and the value V, the output of value iteration.

4.3.1.1 Distributed Global MDP

A centralized MDP may not be feasible if the number of tasks/satellites is large (e.g., 100s/10s). Also, it is difficult to handle the delayed responses. Hence, a distributed approach can be used to find a near optimal solution in real time. In the distributed approach, we will have an MDP for each satellite. Policies of all the satellites are exchanged to come up with an optimal global policy, which is a set of individual policies. The policy of each satellite can be improved iteratively until they converge. The objective is to maximize the global reward. Even though we have separate MDP for each satellite, the whole distributed MDP is solved at the ground station with the knowledge of all the satellites and environmental conditions.



The state contains:

- Time (e.g., t_3^- (start of task 3) or t_3^+ (end of task 3) at orbit 2):
    - Updated after completing a task, performing an action (e.g., wait), or after a communication with a ground station; and
    - Possible time values can be calculated using the task locations and the satellite's orbits.
- Statuses of the tasks {[0, 1, s]}. 0: unserviced. 1: completed, s: another satellite is servicing:
    - Updated after completing a task or after a communication with a ground station;
    - After completing a task, it can go to 1 from any other state;
    - After communicating with the ground station, status s/0 can go to 0 (based on outcome of other satellites), 1 (some other satellite has served), s (probability of serving by another satellite goes above a threshold); and

  - If there is no local MDP, then we can add -1: cannot process (cloud coverage) to status to handle cloud coverage. But, if a local MDP can handle that, we do not need to increase the state space by adding -1 possibility. Instead, we can incorporate that in the transition probability.

- Remaining resources (e.g., memory: 0.1, energy: 0.5, time: 4 (based on duty cycle)).

The possible actions:

- Wait: a_0
- Communicate (downlink and uplink) with the ground station: a^c
- Serve task t: a_t

Rewards:

- Collection reward (applied only to the last state, the state at the end of horizon):
  - Sum of rewards of successfully served tasks; and
  - If we want to give a different reward for a task based on the completion time (i.e., giving more rewards for completing a task within a specified period), then we have to assign the rewards for each outcome, (i.e., immediately after end of action).
- Penalty for using the resources (applied to the states at the end of each orbit, since resources are reset after each orbit):
  - If it is zero, then we may try to use all the allocated resources; and
  - If we can achieve the same collection reward with two different resource requirements, we prefer to pick the one with less resource requirements.

Transition probabilities:

- It can be calculated based on the current (at the time of planning of global MDP) weather forecast and each satellites' sensor success rate; and
- At the ground station communication time steps, the transition probabilities are calculated based on the other satellites' policies.

Regarding deciding if a task is handled by any other satellite, it is not one or zero decision. Rather, we have only a probability for assigning a task based on outcomes of the past actions of all the satellites. Keeping a probability in the state space is hard, hence we can use a threshold (say 0.5) to convert the probability into a one or zero decision.

As mentioned earlier, policies of satellites can be improved iteratively. We need a better initial policy to start up with to find a global optimal policy quickly. An initial policy can be calculated using an open loop solution (e.g., satellite-to-task assignment). Only the transition probabilities of "Communicate" action will change at each iteration.

Clustering discussed under section "Global decision maker without local MDPs", can be used here also to reduce the computational complexity.

4.3.1.2 Local MDP

A local MDP finds a better policy for a satellite for the inter-communication duration. The start state is determined by the outcomes of all the satellites up to the current time and the policies assigned for the other satellites. Additional information that is used to improve the policy provided by the global satellites are:

- Recent weather forecasts provided the ground station; and
- Future weather forecasts by on-board sensors.

Similar to the above global MDP, the state can be defined as:

- Time;
- Statuses of the tasks {[-1, 0, 1, s]}.
    - Except -1, others are the same as the global MDP; and
    - -1 indicates that this task cannot be served (i.e., the probability of success is low due to cloud coverage). Note that we cannot eliminate -1 by incorporating cloud affect in the transition probability. If we do not have on-board sensor, then transition probability handle this issue, but when we have on-board sensors, the transition probabilities are changing, so if we do not add -1, we need to rerun the local MDP to incorporate the changes in the transition probabilities.
- Remaining resources.

In the local MDP, we do not need to consider all the tasks. Only the tasks that are not completed and are within the coverage of the satellite during the specific inter communication interval are considered. Hence, the state space of the local satellite is much smaller than the global MDP.

The possible actions:

- Wait: a_0
- Serve task t: a_t
- Update weather information:
    - After each weather update, the task's status can go from -1 to 0, or 0 to -1.

Rewards:

This is where we have to use the global MDP's output. Somehow, we have to assign rewards for serving each or collection of tasks. If we use only the policy assigned by the global MDP, then we can give rewards only to the task in the optimal policy. Basically, we may need k-best policies to consider different task based on the current weather forecast. The optimal and simplest solution is to use the value $V(s)$ obtained by the value iteration. If we have the reward (value) for reaching a state at the end of the inter-communication interval, then that can used to assign the reward only to the end state, similar to what we have done for global MDP. In the global MDP, end state reward is calculated based on rewards of each task. Here, we assign the reward based on the future action that can be made to achieve the final goal from the end state of the local MDP.

Transition probabilities:

- It can be calculated based on the current (at the start of inter-communication interval) weather forecast and own sensor's success rate.

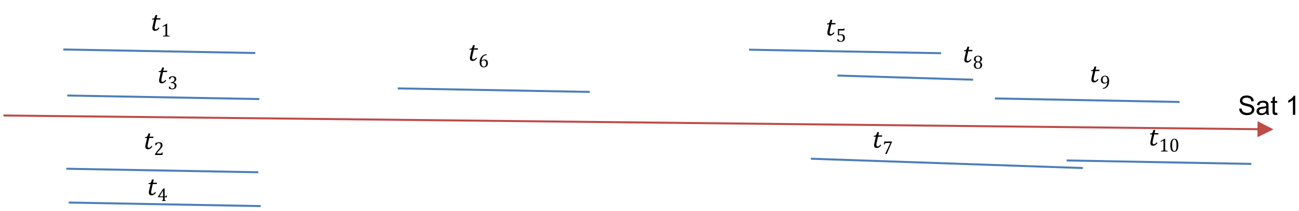
4.3.2 Formulation 2: Global MDP Finds Tasks to Perform for Each Satellite

Global MDP can assign a satellite one of the following:

- Modified priorities of tasks; or
- Rank of tasks;

based on the tasks assigned to other satellites.

Rank of tasks can be assigned with k-best solution approach. However, we need ranks for all the possible future actions at any time. Then, 2-best solution may provide only a single task change. Then, ordering may need to be provided for each conflicting task.

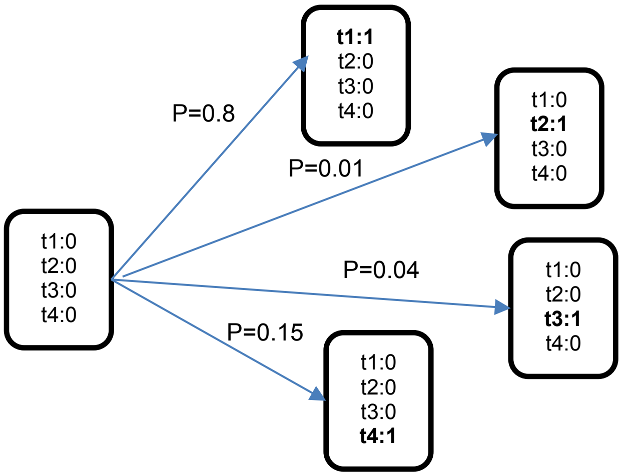


In the above example, if task t_2 is assigned to another satellite and t_1 is assigned to this satellite, then the second option may be task 3 or 4. Similarly, first choice may be task 5 and 9. However, if the second choice is task 8, task 9 or 10 can be followed. If the second choice is task 7, then task 9 or 10 cannot be performed.

Hence, it may be easier to work with modified priorities instead of assigning ranks even though it will increase the state space size. Then, the global MDP just need to assign the modified priority for each task for each satellite. The modified priority is a combination of user-defined priority and the weight given by the global MDP for task-satellite assignment.

In the above example, if we want to assign task 1 to satellite 1 and task 2 to satellite 2, and the second choice for satellite 2 is task 3, then a possible priority assignment may be $[t_1 = 1.0, t_2 = 0.1, t_3 = 0.4, t_4 = 0.6]$. Note that we need to consider the second and third choices of other satellites while deciding the second and third choices of a satellite.

For solving the global MDP, local MDP should return the transition probabilities for given task priorities. Again, we can use local clustering to reduce the possible combinations. Example transition probabilities for cluster one with task 1, 2, 3 and 4 in the previous example is given below.



1.1.1.2 Local MDP

Only the tasks that are not completed (i.e., tasks with priority greater than zero) are considered in the Local MDP.

The state contains:

- Current task;
- Status of the previous task [0, 1]. 0: failed or skipped, 1: completed; and
- Remaining resources.

The possible actions:

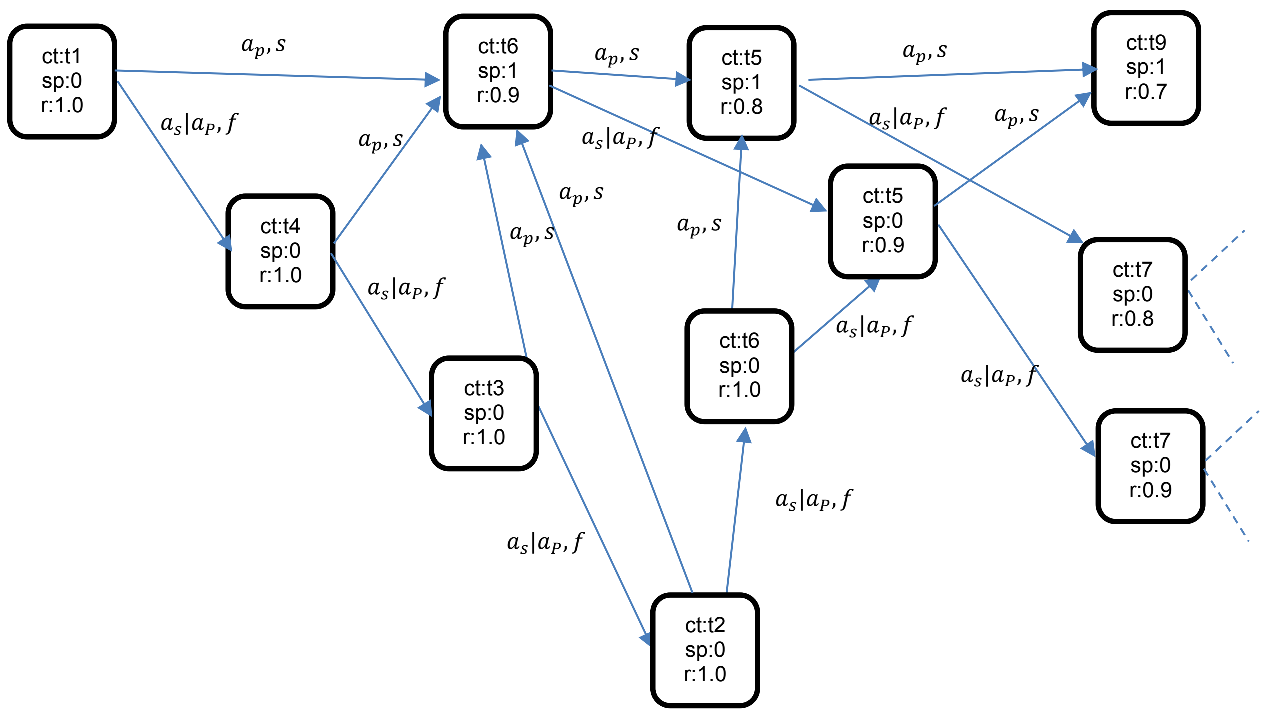
- Skip; and
- Process if feasible (after checking could coverage, etc.).

Rewards:

- Priority (if completed);
- Zero (skip or failed); and
- -1*Infinity (if remaining resources become less than zero).

The above state space model is approximated in the sense that the action on the current task does not consider the cloud coverage of future tasks.

The tasks can be ordered based on the start time and the priority. A sample state space based on example tasks considered above:



In the above figure:

- ct – current task;
- sp – status of previous task;
- r – remaining resources;
- a_p, s – action: process if feasible, outcome: success; and
- $a_s|a_P, f$ - action skip, or action: process if feasible, outcome: failure.

Local MDP does not need to handle possible new tasks since local MDP consider the actions from one communication time to next communication time during which new tasks appeared at the ground station level will not be available to the satellite. Hence, new tasks should be handled by the ground station (i.e., global MDP).

1.1.1.3 Global MDP

The objective is to assign modified priorities to tasks for each satellite.

1.1.1.4 Approach 1

Global MDP proposed under "Global decision maker without local MDPs" can be used to find the best satellite to task assignment under the assumption that the environmental or other conditions that may affect performing a task may not change. Then, we can use another optimization algorithm to assign modified priorities to the tasks for each satellite based on the first choice obtained by the Global MDP. One possible approach is to exclude all the tasks assigned with the first choice and solve an assignment problem or another MDP with remaining tasks. We can continue this process for k (a predefined, or until the remaining tasks set becomes zero) iteration by excluding the tasks already picked. Then, we can assign the modified priorities based on the tasks' assigned iteration number.

1.1.1.5 Approach 2

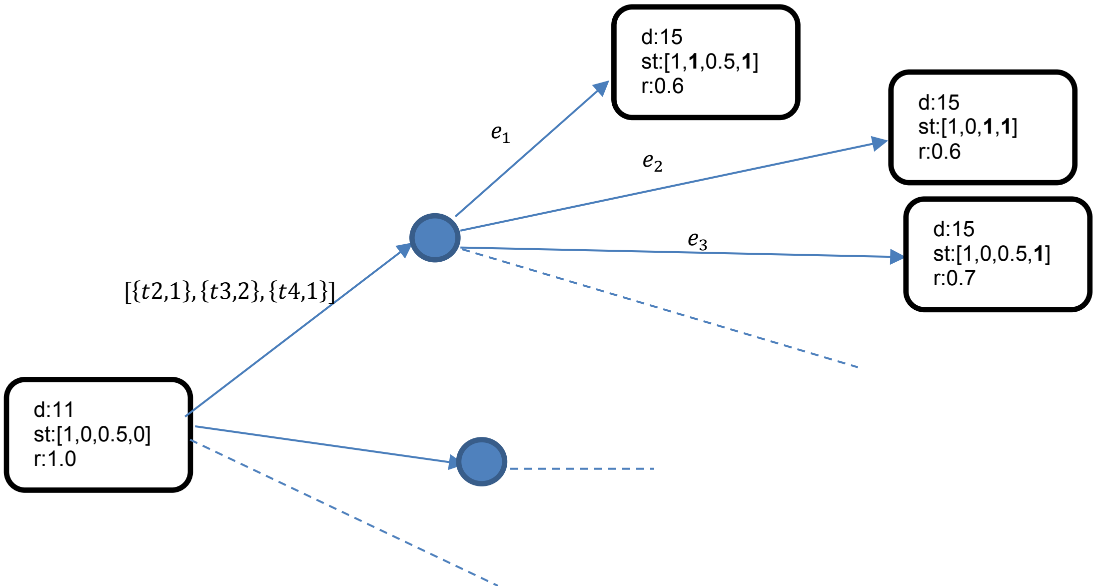
If we want to consider this problem as a single MDP, the problem should be handled at the download level rather than at the orbit level. We can assume that the number of downloads and their times are known.

State:

- Download number d
  - Orbit number o
  - Satellite number s
- State of task {0,0.5,1}
  - 0: need to scan;
  - 0.5: a satellite is processing:
    - Set to this only the assigned task's rank/priority is 1. If we want to include other ranks, then we may need to have more state levels.
  - 1: done.
- Satellite resource level {0, 0.1, ..., 10}
  - Satellite failure can be handled by setting this to zero.

Actions (for satellite s):

- task
- rank/priority $\{1, 2, 3, 4, .., M\}$
  - No need to have rank 0 since those tasks will not be included in the action set; and
  - Priority numbers can be converted to priority values before using with local MDP.



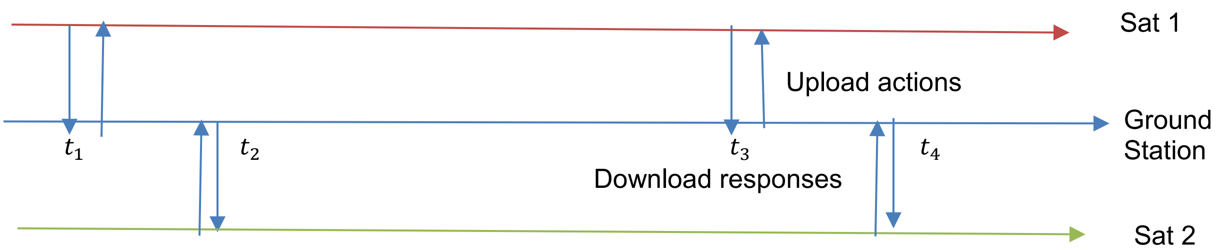
We need to handle the delayed response. In the above example, we are going to have three more downloads before we get the download from the same satellite. Then, there will be three more state changes before we come to $d = 15$ from $d = 11$. Augmented MDP with much larger state space S × A is a solution, but not feasible due to huge computational complexity.

5 A PROBLEM FORMULATION FOR AGILE SATELLITES:

Assumptions:

- Download /upload times of each satellite is predefined and known; and
- Based on above times and the satellite orbits, the set of possible actions during each communication intervals are known.

Then, we can have a global MDP, and local MDP for each satellite.



If the horizon is from t_1 to t_{10}, the global MDP will run from t_1 to t_{10}, and the local MDPs will run for each time segment. For example, a local MDP for Sat 1 will run from t_1 to t_3, which can be used to decide the internal actions of Sat1, and to get the reward for assigned actions for the global MDP.

Then, the global state can contain:

- Time step t_t
- State of task a
  - {sat s is processing, done, failed/not assigned}
  - We need to consider the satellites that can perform the task a during t_t to next download time
- Remaining resources of Sat s
  - If the energy is limited, we can discretize the energy levels (e.g., 10 levels)

If the number of satellites is 3 and the number of tasks is 10, with the above state space, at each time step, we may have $5^{10} * 10^3 = 9.7656e + 09$, which is harder to handle.

5.1 An approximation

- First, allocate tasks to satellites:
  - Use MDP to find rewards for each assignment.
- Then, solve each local MDP:
  - Still, state space can be huge.

        - For example, if we allocate 10 tasks for a satellite, number of states (state = {state of task (done, failed/not done), remaining resources (10 levels)} will be $2^{10} * 10 = 10240$
    - To reduce the state space, we can order the tasks:
        - Another optimization problem; and
        - Assume only one task can be performed at a time.
    - Then, the state contains:
        - Current task (ordered task, e.g., second task);
        - Remaining time steps for current task;
        - Tasks failed:
            - Since the failed tasks will not be tried again by the same satellite, future action will not depend on the failed tasks. So, we can remove this from the state.
        - Remaining resources; and
        - Time step.
    - Possible actions:
        - Move to next task:
            - If the "Remaining time steps for current task" is not equal to zero, the current task is considered failed.
        - Process the same task.
    - Failed actions can be sent to ground stations, which can redirect them to second best satellite considering the current time step, sat position, and task region;
    - In order to handle failed tasks by other satellites and possible new tasks, each local MDP should support new tasks:
        - A possible new task should be added to the state space:
            - Priority/weight;
            - Number of time steps required; and
            - Coverage time step range.
- Global MDP will handle the new and failed tasks assignment to Satellites.

5.1.1 Task to Satellite assignment

The objective is

$$\min \sum_{t=1}^{T} \sum_{s=1}^{S} a(t,s)c(t,s)$$

Subject to

$$\sum_{s=1}^{S} a(t,s) = 1, \forall\, t$$
$$\sum_{t=1}^{T} a(t,s)r(t,s) \leq\ R_s, \forall\, s$$
$$a(t,k) \in \{0,1\}$$

Need to add more constraints to:

- Dependent tasks (overlapping or consecutive time slots); and
- Overloading one satellite.

It may be easier to combine task to satellite assignment and the task ordering to incorporate additional constraints.

5.1.2 Task-Satellite assignment with Timestep Assignment

Assume the time horizon is divided into K timesteps. If the unknown variable is $a(s,t,k)$, the assignment of task t to satellite s and start the task at timestep k, the objective is

$$\arg\min \sum_{s=1}^{S} \sum_{t=1}^{T} \sum_{k=1}^{K} a(s,t,k)c(s,t,k)$$

Subject to

$$a(s,t,k) \in \{0,1\}$$
$$\sum_{s=1}^{S} \sum_{k=1}^{K} a(s,t,k) = 1, \forall\, t$$
$$\sum_{t=1}^{T} \sum_{l=0}^{K_t^s} a(s,t,k-l) \leq 1\ \forall\, s,k$$
$$\sum_{t=1}^{T} \sum_{k=1}^{K} a(s,t,k)R(s,t) \leq R_s, \quad \forall\, s$$

Where

K_t^s – time steps required to process task t at satellite s

Note that the start time k indicates just the start time. For example, if a task starts at k=5 and takes 3 timesteps to complete, then $a(s,t,5) = 1$ and $a(s,t,6) = 0$. Also, $a(s,:,6)\ =\ 0$.

The cost $c(s,t,k)$ can be assigned based on the possible error (RMSE for tracking, coverage quality for surveillance, etc.) result on performing the task t by satellite s from k to $k + K_t^s$.

We need to see how to incorporate uncertainty in the outcome (MDP) in assigning the cost.

The resource constraints can be expanded to include memory and energy constraints separately. Also, if the memory can be cleared/reset after a download, that can also be incorporated easily since we have k in the unknown variable.

Possible new tasks are not considered in the above assignment, but we may add some reward for leaving few empty time slots.

5.1.3 Local MDP

From the solution of $a(s,t,k)$, we can find the task ordering for each satellite. The actual start time of each task can be found using the local MDP. The task's start timestep obtained by the above assignment is used only for the ordering.

The state is:

- Current task (ordered task, e.g., second task), i_t
- Remaining time steps for current task, k_t
- Remaining resources, r
- Time step, k

The i-th state at timestep k can be given as $S_i^k = \{i_t, k_t, r, k\}$. For example, if the second task is started at time step 5, and that task requires 3 timesteps, the two of the possible sequences of states are

$\{2,3,0.8,5\}, \{2,2,0.7,6\}, \{2,1,0.7,7\}, \{2,0,0.6,8\}$

$\{2,3,0.8,5\}, \{2,2,0.7,6\}, \{2,2,0.7,7\}, \{2,1,0.6,8\}, \{2,0,0.6,9\}$

Actions:

- Start (or wait for, if the task is not in the coverage yet) next task:
  - If the current k_t is greater than zero, the previous task is failed.
- Continue current task.

Need to add more to the state to handle possible new states.

5.1.4 Global MDP

The purpose of global MDP is to assign new and failed tasks to satellites. We need to consider all the possible new/failed tasks and their combinations. MDP states of all the satellites also need to be considered. An alternative would be to use assignment instead of MDP here also.

6 MDP

Our objective is to choose optimal actions over a time horizon *H* such that the expected cumulative cost is minimized.

The cumulative cost is given by

$$C_H = E(\sum_{k=0}^{H-1} c(\, s_k, a_k))$$

where s_k is the state at time *k* and a_k is the action at time *k*. The cost of taking action a_k in state s_k is given by $c(s_k, a_k)$.

The optimal policy, a sequence of actions, is given by

$$\pi^* = \arg\max_{\pi} E(\sum_{k=0}^{H-1} c(s_k, a_k)|\pi)$$

where $a_k = \pi(s_k)$.

6.1 State tree

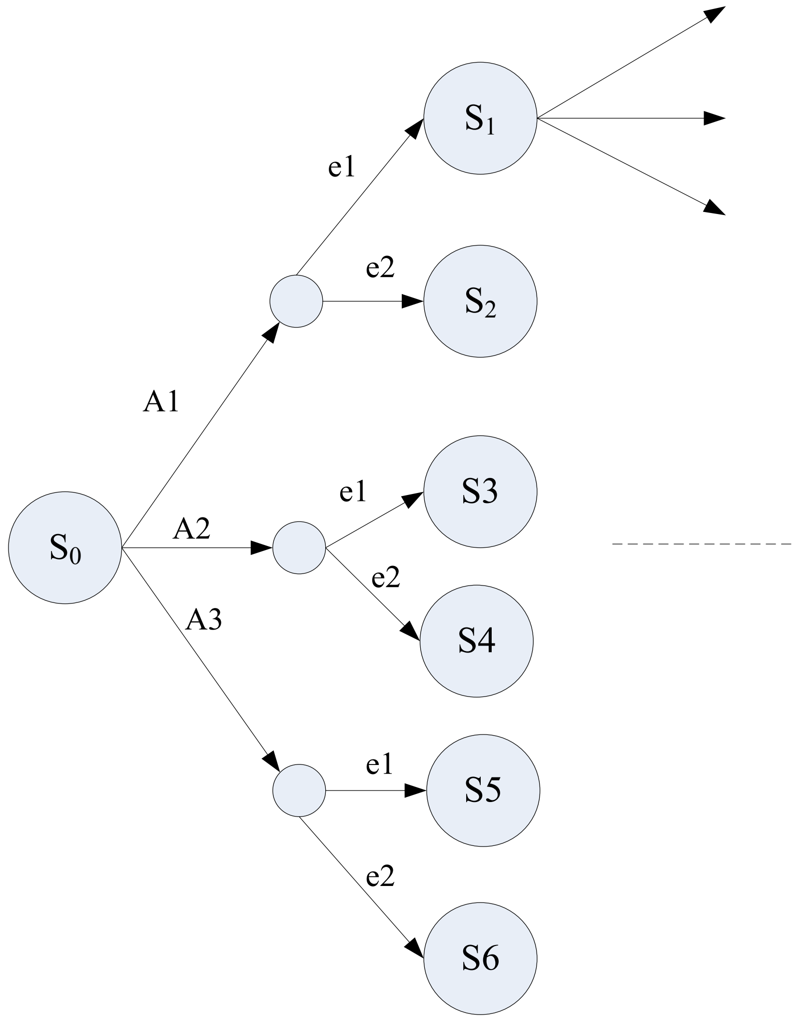


Figure 6–1: A sample state tree with actions and events

6.1.1 Scalarizing function

With multiple objectives, finding all Pareto-optimal solutions can be difficult and time consuming for the problems where the number of Pareto-optimal value functions is exponential in the number of states. Mostly, a compromise solution achieving interesting trade-offs between objectives is desired. A scalarizing function can be used to form a single objective to find a compromised solution:

$$C = \phi(C\_1, C\_2, \ldots, C_n)$$

The choices for the scalarizing functions for cost (reward) are:

- Weighted sum:

$$C = \sum_{i=1}^{n} w_i s_i C_i$$

where w_i and s_i are the weight and scaling factor for the cost i, respectively. The scaling factors can be used to bring the costs to a common unit and the weights can be used to give different priority to different objectives.

- Maximum (minimum) value:

$$C = \max(w_1 s_1 C_1, w_2 s_2 C_2, \dots, w_n s_n C_n)$$

- Maximum (minimum) regret:

$$C = \max(w_1 s_1 (C_1 - I_1), w_2 s_2 (C_2 - I_2), \dots, w_n s_n (C_n - I_n))$$

Where I_i is the ideal value for cost *i*.

- Weighted regret:

$$C = \sum_{i=1}^{n} w_i s_i (C_i - I_i)$$

6.2 Solution algorithms

6.2.1 Value iteration

In this method, the optimal infinite horizon policy is found using a sequence of finite horizon value functions. The steps of the value iteration algorithms are:

- Initialize i=0 and V0(s) = 0 for all $s \in S$
- While $\max_{s \epsilon S} |V_{i+1}(s) - V_i(s)| > \epsilon$,

$$\max_{s \in S} (V_{i+1}(s) - V_i(S) > \epsilon$$

calculate $V_{i+1}(s)$ as follows and increment i:

$$V_{i+1}(s) = \max_{a \in A} \; [R(s,a) + \gamma \sum_{s' \in S} T(s,a,S') V_i(s')$$

6.2.2 Policy iteration

This method consists of two stages: policy evaluation and policy improvement. The steps of a policy iteration algorithm are:

- Initialize $\pi_o(s) = a$ for all $s \in S$; $a \in A$ is an arbitrary action;
- Policy evaluation: calculate the value of a policy π_i

$$V^{\pi_i}(s) = R(s, \pi_i(s)) + \gamma \sum_{s' \in S} T(s, a, S') V^{\pi_t}(s')$$

- Policy improvement:
  - Compute Q-function:

$$Q_{i+1}(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s, a, S') V^{\pi_t}(s')$$

  - Improve the policy:

$$\pi_{i+1}(s) = arg\max_{a \in A} Q_{i+1}(s, a) \text{ for all } s \in S$$

- Repeat the above iteration until the policy does not change anymore.

Policy iteration tends to converge faster than value iteration, but the computation at each step is more.

7 ALGORITHM INVESTIGATION, IMPLEMENTATION AND TESTING

The resource management framework consisting of global MDP and local POMDP was implemented using C++. The software framework was designed such that it is highly parameterized and different scenarios with different number of satellites, types of tasks and constraints can be simulated just by modifying XML configuration files.

The salient features of the global and local resource management software framework are as follows:

- Global MDP decision maker with local POMDP:
    - See Section 2.3.1 in this document.
- Global MDP:
    - Distributed (iterative) MDP:
        - Optimal policy for a satellite is found assuming other satellites' policies are fixed and known; and
        - Used few iterations to improve the solution.
    - Custering is used to reduce the computational load and memory requirements.
- Local POMDP:
    - Consider possible weather update:
        - New failure probabilities.
    - Global MDP's value function at the end time is used as the reward for end states.

The steps for invoking the software are as follows:

- Prepare input XML files:
    - We can specify multiple projects.
- Run the code after changing the project path:
    - The default project is "input/project.xml".
- Outputs will be saved under "output" folder:
    - Different projects can have different output folders; and
    - Output is saved only if "enable-save" yes under project settings.
- Matlab code "matlab\plot_scenario.m" can be used to view the tasks.

The specification of different scenarios requires the modification or customization of the following XML files:

- project.xml
- satellites.xml
- tasks.xml
- ground_stations.xml
- environment.xml
- scheduler.xml
    - global_mdp.xml
    - local_mdp.xml
    - mdp_xml

A representative satellite XML, where satellite orbits are specified using two-line element (TLE) sets, is given below:

```xml
<satellite name="RCM2">
    <orbit type="tle">
        <!-- constellation, tle-existing, tle -->
        <tle>
            <line_1>1 99999U          15182.00000000  .00000125  00000-0  12143-4 0 00008</line_1>
            <line_2>2 99999 097.7544 188.7205 0002363 359.8062 142.7546 14.92589026000017</line_2>
        </tle>
    </orbit>
    <resources>
        <limit>
            <memory>100</memory>
            <energy>20</energy>
            <duty-cycle>0.2</duty-cycle>
        </limit>
        <consumption-per-hour-task>
            <memory>10</memory>
            <energy>1.5</energy>
        </consumption-per-hour-task>
        <energy-reset>orbit</energy-reset>
    </resources>
    <coverage>
        <roll-min>-20</roll-min>
        <roll-max>20</roll-max>
    </coverage>
    <ground-station-connection>
        <!-- need to discuss -->
        <ground-radius-km>500</ground-radius-km>
    </ground-station-connection>
    <sensor>
        <!-- what parameters do we need to add -->
        <enery-for-1deg-sensor-rotation>0.01</enery-for-1deg-sensor-rotation>
    </sensor>
</satellite>
```

A representative task XML file, where a task could be a strip of a request that is specified with the start point and the length, is given below:

```xml
<task name="task2">
    <region>
        <start-point latitude="45.5" longitude="-76" altitude="0"/>
        <length-in-km>100</length-in-km>
    </region>
    <reward>
        <complete>10</complete>
    </reward>
</task>
```

A representative ground station XML file, where its location is specified, is given below:


```xml
<ground-stations>
    <station name="Gatineau">
        <latitude>45.5853133</latitude>
        <longitude>-75.8107334</longitude>
        <altitude>0</altitude>
    </station>
    <station name="Inuvik">
        <latitude>68.3193875</latitude>
        <longitude>-133.5528679</longitude>
        <altitude>0</altitude>
    </station>
    <station name="PrinceAlbert">
        <latitude>53.4354707</latitude>
        <longitude>-105.185193</longitude>
        <altitude>0</altitude>
    </station>
```

A representative environment XML file, where failure probabilities due to factors such as weather or communication failures for different regions and time windows are specified, is given below:

```
<environment>
    <failure-probability>
        <default>0.05</default>
        <region>
            <polygon>
                <point latitude="45.00" longitude="-77" altitude="0"/>
                <point latitude="46.00" longitude="-77" altitude="0"/>
                <point latitude="46.00" longitude="-75" altitude="0"/>
                <point latitude="43.00" longitude="-75" altitude="0"/>
            </polygon>
            <default>0.08</default>
            <time-slot>
                <start-time utc="no">
                    <year>2014</year>
                    <month>12</month>
                    <day>16</day>
                    <hour>06</hour>
                    <minute>00</minute>
                </start-time>
                <end-time utc="no">
                    <year>2014</year>
                    <month>12</month>
                    <day>16</day>
                    <hour>09</hour>
                    <minute>00</minute>
                </end-time>
                <value>0.2</value>
            </time-slot>
```

A representative scheduler XML file, where time windows, minimum step sizes and paths for global MDP and local POMDP configuration files are specified, is given below:

```
<scheduler>
    <start-time utc="no">
        <year>2014</year>
        <month>12</month>
        <day>16</day>
        <hour>05</hour>
        <minute>00</minute>
    </start-time>
    <time-horizon-in-hours>12</time-horizon-in-hours>
    <minimum-time-step-in-seconds>10</minimum-time-step-in-seconds>

    <mdp>
        <global-mdp-file>global_mdp.xml</global-mdp-file>
        <local-mdp-file>local_mdp.xml</local-mdp-file>
    </mdp>
</scheduler>
```

A representative global MDP file, where resource constraint discretization and clustering options are specified, is given below:

```
<globalmdp>
    <use-clustering>1</use-clustering>
    <mdp-file>mdp.xml</mdp-file>
    <distributed>
        <maximum-iterations>10</maximum-iterations>
        <resource-discretization-levels>10</resource-discretization-levels>
    </distributed>
    <ignore-resource-constraint>1</ignore-resource-constraint>
    <!-- can be used for testing -->
</globalmdp>
```

We also developed a task output viewer to visualize the output of the resource manager. A representative output, where the blue lines mark the task strips, is given below:

Figure 7–2: Representative output

Satellite trajectories were generated using TrackGen's ISR360 software for simulation, target tracking and sensor fusion. The code for this task visualization module is also given in the repository. A representative set of satellite trajectories is given below:

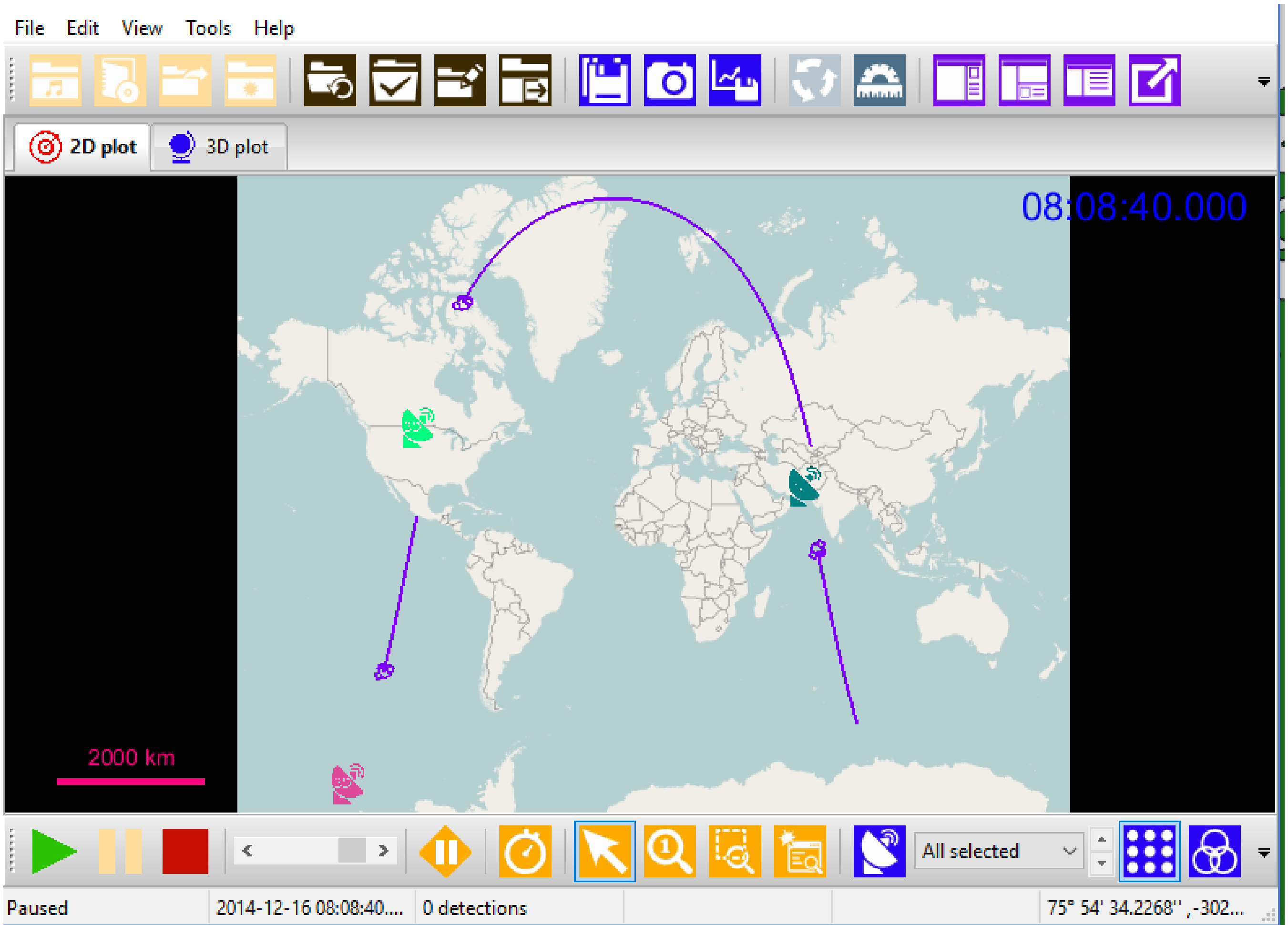


Figure 7–3: Representative set of satellite trajectories

8 EVALUATION AND COMPARATIVE PERFORMANCE ASSESSMENT

The proposed framework and the software were tested on different scenarios with different parameters.

With 3 satellites and 5 different tasks, the global MDP and local POMDP solutions took 2.1 and 4.2s, respectively. Clustering was used to reduce the number of states substantially. The figures below show the numbers of states in the global MDP and local POMDP formulations:

```
+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
Global MDp results:
Number of clusters: 7
--------------------------------------------------------------------
Cluster 1 results:
Tasks: 1,2,3
*********************************************************************
Satellite RCM1 results:
Number of states:4753
MDP value of root node:8.006
*********************************************************************
Satellite RCM2 results:
Number of states:2377
MDP value of root node:0.003
*********************************************************************
Satellite RCM3 results:
Number of states:2971
MDP value of root node:0.004
--------------------------------------------------------------------
Cluster 2 results:
Tasks: 4,5
*********************************************************************
Satellite RCM1 results:
Number of states:1486
MDP value of root node:23.756
*********************************************************************
Satellite RCM2 results:
Number of states:793
MDP value of root node:0.003
*********************************************************************
```

Figure 8–4: Numbers of states in the global MDP formulations

```
********************************************************************
Satellite RCM3 results:
Number of states:463
MDP value of root node:19.955
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Excecution time for global MDP: 2.046s
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
local MDp results:
Satellite: RCM1
Start time: 2014-12-16 06:38:50
End time: 2014-12-16 07:47:30
Tasks: 1,2,3,4,5
Number of states:78849
MDP value of root node:31.76
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

Excecution time for local MDP: 4.193s
Press 'Enter' to terminate.
```

Figure 8–5: Numbers of states in the local POMDP formulations

More extensive experiments with comparison to open-loop formulations will be conducted to present the results of this work in a journal publication.

9 BIBLIOGRAPHY

D. Akselrod and T. Kirubarajan, "Markov Decision Process-Based Resource and Information Management for Sensor Networks," in Sensor Networks: Where Theory Meets Practice, (G. Ferrari, ed.), Springer, Germany, pp. 167-218, 2009.

T. Brehard, P. A. Coquelin, E. Duflos and P. Vanheeghe, "Optimal Policies Search for Sensor Management: Application to the ESA radar," *2008 11th International Conference on Information Fusion*, Cologne, 2008, pp. 1-8.

R. C. Chen and K. Wagner, "Constrained Partially Observed Markov Decision Processes for Adaptive Waveform Scheduling," *2007 International Conference on Electromagnetics in Advanced Applications*, Torino, 2007, pp. 454-463.

E. Chong, "Adaptive Sensor Management: POMDP Approximation Methods,", ARO-MURI Adaptive Sensing and Waveform Design Workshop, Aug. 2-3, 2005.

V. Krishnamurthy and D. V. Djonin, "Structured Threshold Policies for Dynamic Sensor Scheduling—A Partially Observed Markov Decision Process Approach," in *IEEE Transactions on Signal Processing*, vol. 55, no. 10, pp. 4938-4957, Oct. 2007.

V. Krishnamurthy and D. V. Djonin, "Optimal Threshold Policies for Multivariate POMDPs in Radar Resource Management," in *IEEE Transactions on Signal Processing*, vol. 57, no. 10, pp. 3954-3969, Oct. 2009.

M. Rezaeian and B. Moran, "Constrained Multi-Object Markov Decision Scheduling with Application to Radar Resource Management," 2010 13th International Conference on Information Fusion, Edinburgh, 2010, pp. 1-8.

J. Wang, E. Demeulemeester, D. Qiu and J. Liu, "Exact and Inexact Scheduling Algorithms for Multiple Earth Observation Satellites Under Uncertainties of Clouds," Computers and Operations Research, Vol. 74, Issue C, pp. 1-13, October 2016.

DOCUMENT CONTROL DATA

*Security markings for the title, authors, abstract and keywords must be entered when the document is sensitive

1. ORIGINATOR (Name and address of the organization preparing the document. A DRDC Centre sponsoring a contractor's report, or tasking agency, is entered in Section 8.) Thales Research & Technology (TRT) Canada 1405, boul. du Parc Technologique Québec, QC G1P 4P5	2a. SECURITY MARKING (Overall security marking of the document including special supplemental markings if applicable.) CAN UNCLASSIFIED 2b. CONTROLLED GOODS NON-CONTROLLED GOODS DMC A

3. TITLE (The document title and sub-title as indicated on the title page.)

Development of a Joint Intelligence and Information S&T Capability Task Authorization 22—Deployable Intelligence Source Collection Value Optimizer (DISCOVER)—Multi-Satellite Collection Scheduling: Closed-Loop Collection Tasking (Multiple Episodes)—Implementation

4. AUTHORS (Last name, followed by initials – ranks, titles, etc., not to be used)

Tharmarasa, R.; Florea, M.; Kirubarajan, T.

5. DATE OF PUBLICATION (Month and year of publication of document.)	6a. NO. OF PAGES (Total pages, including Annexes, excluding DCD, covering and verso pages.)	6b. NO. OF REFS (Total references cited.)
March 2018	40	8

7. DOCUMENT CATEGORY (e.g., Scientific Report, Contract Report, Scientific Letter.)

Contract Report

8. SPONSORING CENTRE (The name and address of the department project office or laboratory sponsoring the research and development.)

DRDC – Valcartier Research Centre
Defence Research and Development Canada
2459 route de la Bravoure
Québec (Québec) G3J 1X5
Canada

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)	9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)
05da—Joint Intelligence Collection and Analysis Capability (JICAC)	W7701-125076/001/QCL

10a. DRDC PUBLICATION NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)	10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)
DRDC-RDDC-2018-C189	

11a. FUTURE DISTRIBUTION WITHIN CANADA (Approval for further dissemination of the document. Security classification must also be considered.)

Public release

11b. FUTURE DISTRIBUTION OUTSIDE CANADA (Approval for further dissemination of the document. Security classification must also be considered.)

12. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Use semi-colon as a delimiter.)

Multi-Satellite Scheduling

13. ABSTRACT/RÉSUMÉ (When available in the document, the French version of the abstract must be included here.)

Closed-loop collection tasking does explicitly account for information feedback (e.g., observation outcome) interleaving task planning and task plan execution. Environment assumptions are typically characterized by:

- uncertainty,
- dynamic (on-line) conditional (real-time) task planning (sequential decision),
- multiple tasking episodes, and,
- a high tempo of operations (state transition rate /events) requiring responsiveness.

Dynamic tasking may be appropriate when operating under uncertainty, latency conditions or high operations tempo/state transition rate (e.g., time-dependent communication and decision cycle, observation outcome/feedback, resource availability/failure, urgent task occurrence, delayed observation and information feedback, information-sharing, weather conditions).

Through this document, we explore adaptive and contingency planning/scheduling concepts to the dynamic multi-satellite collection scheduling problem. We have also developed a new adaptive scheduling decision problem model properly dealing with state transitions (e.g., exogenous event, new task, feedback information, resource availability/failure, urgent observation requests) while planning and executing tasks and trading-off run-time and solution quality.

La tâche de collecte en boucle fermée tient en compte de façon explicite du retour d’information (e.g. le résultat de l’observation) entrelaçant la planification des tâches avec l’exécution du plan. Les hypothèses de l’environnement sont caractérisées habituellement par:

- incertitude,
- planification des tâches (décision séquentielle), conditionnelle (temps-réel) et dynamique (enligne)
- épisodes multiples d’assignation de tâches, et,
- une fréquence élevée d’opérations (transition d’états) qui nécessite une réactivité accrue.

L’assignation de tâches en mode dynamique peut être peut s’avérer très utile lorsqu’on opère sous incertitude, en conditions nécessitant des grands délais ou des grands temps de réponses, et lorsque la fréquence des d’opérations est élevée (e.g., communications et cycle décisionnel qui dépendent du temps, retour des observations ou du résultat, disponibilité ou défaut des ressources, rajout des tâches urgentes, retour d’observations et d’information en retard, partage d’informations, conditions météo).

À travers ce document, nous explorons des nouveaux concepts de planification adaptive et d’urgence pour le problème de planification dynamique de collections multi-satellites. Nous avons aussi developpé un nouveau modèle pour le problème de décision pour une planification adaptative qui traite correctement avec les transitions d’états (e.g., événements exogènes, nouvelles tâches, retour d’information, disponibilité/défaut des ressources, requêtes urgentes d’observation) lors de la planification et l’exécution des tâches, en trouvant un équilibre entre le temps de calcul et la qualité de la solution.