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—Open-Loop Collection Tasking

Mihai Florea
Emmanuel Giasson
Thales Research & Technology (TRT) Canada

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

PSPC Contract Number: W7701-125076/001/QCL
Technical Authority: Jean Berger, Defence Scientist
Contractor's date of publication: September 2018

Defence Research and Development Canada

Contract Report
DRDC-RDDC-2018-C168
September 2018

Canada

IMPORTANT INFORMATIVE STATEMENTS

This document was reviewed for Controlled Goods by Defence Research and Development Canada using the Schedule to the *Defence Production Act*.

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
Open-Loop Collection Tasking

Contract No.: W7701-125076/001/QCL

Document Control No.: 2066C.022-REP-01-OLCT Rev. 02

Date: 26 January 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

Open-Loop Collection Tasking

Contract No.: W7701-125076/001/QCL

Approved by:

Steeve Côté
Project Manager, Thales Canada Inc.

REVISION HISTORY

Date	Rev.	Description	Author
31 May 2017	01	Initial release	Mihai Florea
26 January 2018	02	Final release	Emmanuel Giasson

TABLE OF CONTENTS

LIST OF FIGURES

Page

LIST OF TABLES

Page

LIST OF ANNEXES

Annex A	Generated test data
Annex B	Wang test data
Annex C	Solver comparison on Wang data sets

LIST OF ACRONYMS AND ABBREVIATIONS

A

AFSCN	Air Force Satellite Control Network
AOI	Area of Interest

C

CFG	Coarse- and Fine-grained
CP	Collection Plan, Collection Plan
CSIAPS	Canadian Satellite Imagery Acquisition Planning System
CT	Collection Task

D

DISCOVER	Deployable Intelligence Source Collection Value OptimizER
DSN	Deep Space Network

E

EO	Electro-Optical

G

GATHER	Genetic Algorithm-based collecTion scHedulER

I

ILP	Integer Linear Programming
IR	Infrared
IRM	Intelligence Requirements Management

L

LS	Local Search

M

MOE	Measure of Effectiveness
MOP	Measure of Performance
MOSAIC	Mixed/Multi Overlapping Sub-Area composition Coverage
MSICSP	Multi-Satellite Intelligence Collection Scheduling Problem
MY-PICC	MYopic Planning-based Image aCquisition heuristiC

N

NIIRS	National Imagery Interpretability Rating Scales

P

PV	Plain-Value

Q

QoS	Quality of Service
QUEST	QUadratically constrained cplEx-based Solver (Scheduler) Technology

R

RE-CVEST	REward Collection Value cplEx-based Scheduler Technology
RFI	Request for Information

S

SAR	Synthetic Aperture Radar
SCBH	Set Covering and Blocks Holes
SSR	Solid-State Recorder

T

TW	Time-Weighted

V

VCSP	Valued Constraint Satisfaction Problem
VoI	Value of Information

This report is prepared by Thales Canada as a delivery for the Subtask 6.1 – “Open-loop collection tasking - Multi-satellite collection task scheduling – cycle I” of the Task Authorization 22 – “Deployable Intelligence Source Collection Value Optimizer (DISCOVER) – multi-satellite collection scheduling” from the contract W7701-125076/001/QCL.

1 DATA MODELS

1.1 Collection Task (CT) Data Model (1st draft)

The collection task data model is designed to support the multi-satellite imagery collection scheduling problem, as described in "*Implement Plan Sched 2016-2017 v2 16 Dec 16.docx*". Data is structured in a tuple <*R, C, Sppt, Obj, A, M, CONS* > where:

- *R*: set of requests;
- *C*: set of collection assets (collectors);
- *Sppt*: set of supporting resources – typically, ground stations;
- *Obj*: collection tasking objective (open-loop multi-satellite collection scheduling);
- *A*: Action space/set; and
- *M:* Matches between tasks and collection sensor package capabilities.

1.1.1 Design Overview

All Java code related to the data model is packaged under **`ca.gc.rddc.discover.commons.*`** packages, in the **`DiscoverCommons`** module. Until real data is available (namely from CSIAPS), the following design was observed. It is expected to change to meet actual data interface once the latter is ready:

- There is a set of classes for each type of object/container required by the scheduling problem:
    - Those are detailed in the remainder of this data model section; and
    - A single class, **`DataSet`** (see 1.1.3 *Data set*), serves as the data provider and provides access to all others, exception for the objective function (see 1.1.8 *Collection tasking objective*).
- A set of interface classes offers a read-only access to the external data (whether accessed directly through some yet-to-be-defined mechanism, or first converted to an in-memory representation):
    - Their names and the data they give access to (while not containing it) follow the data model described in this section; and
    - See **`ca.gc.rddc.discover.commons.dataaccess`** and associated generated Javadoc.
- A parallel set of classes was developed, at least temporarily, to implement each of the above interfaces with an in-memory read-writable representation:
    - See **`ca.gc.rddc.discover.commons.inmemory`** and associated generated Javadoc;
    - To allow the processing of very large data sets, the well-known performance-oriented (speed and memory footprint) **`fastuil`** library was chosen to implement internal maps and sets. See http://fastutil.di.unimi.it; and
    - To import data from an external source (format used by Wang et al. [1]), a separate package contains required class to convert into DISCOVER data model. See **`ca.gc.rddc.discover.commons.externaldata.wang`**.

- Several state-less data computations and manipulations were gathered together:
    - They are used by all DISCOVER solvers to make sure that the same concepts are executed the same way across solvers;
    - Some of them may eventually be offered by an external service instead of being computed/estimated from the data; and
    - See `ca.gc.rddc.discover.commons.datautils` and associated generated Javadoc.

Given the temporary and/or likely-to-change-shortly nature of this design regarding external data access, available work effort was limited (in agreement with all involved parties) such that:

- The `dataaccess` and `inmemory` packages have little to no logic. In-code documentation and unit tests cover all classes but are pretty lean:
    - Completion is postponed until the design is stable; and
    - This document nevertheless provides necessary documentation.
- Other packages are documented and covered by proper unit tests.

1.1.2 How to read this section

Color/font key for the remainder of this data model section:

- Current implementation:
    - `name: type` : implementation details, and corresponding original text from Implementation Plan.
- Original text from Implementation Plan, not implemented yet; and
- Data already left out in the original *implementation Plan* (for future tasks/versions).

Data types are inspired from Java and JSON types to allow for easy conversion between data sources. For the sake of simplicity and interoperability, types are limited to the following base types to compose other types:

- `int`: a 32-bit signed integral value;
- `double`: a 64-bit floating point value (IEEE 754 compliant);
- `string`: any text;
- *`type[]`*: a collection of items of type *`type`*. Actual underlying structure (linked list, array, etc.) is irrelevant, unless otherwise specified;
- *`key_type -> value_type`*: a collection that is specifically a mapping from keys to values. Unless otherwise specified, keys are assumed to be unique and sorting (or lack thereof) is irrelevant;
- *`type in [n, m]`*: a value of type *`type`*, bounded by `n` and `m` (`n` and `m` are included/excluded depending on the bracket orientation). Bounding is logical only, per specification, and is not enforced in the code; and

- ***named_type***: a composite type named ***named_type***, with its decomposition:
    - **member1: type1**;
    - **member2: type2**; and
    - …

1.1.3 Data set

DataSet:

- **requestData: RequestData** : see 1.1.4 *Set of requests (tasks)*;
- **collectorData: CollectorData** : see 1.1.5 *Set of collection assets (collectors)*;
- **matchData: MatchData** : see 1.1.6 *Sensor–task matchmaking*;
- **constraintData: ConstraintData** : see 1.1.7 *Constraints set*; and
- **predefinedTaskPlans: String -> TaskPlan[]** : see 1.1.9 *Pre-Defined Task Plans*.

Other parts of the tuple (from the imagery collection scheduling problem) that are not yet implemented are listed in 1.1.8 Collection tasking objective.

1.1.4 Set of requests (tasks)

An imaging request (or order) refers to a specific area interest (AOI) to be surveyed or covered by a space-borne sensor and can be de-decomposed in a single (primitive) or multiple (compound) survey tasks through a task generation process. An AOI is either defined as a spot (small target) or a polygon referring to a primitive and compound task respectively. A polygon can be decomposed in multiple strips corresponding to satellite swaths intersecting the AOI during a given orbit. A strip defines a simple or primitive task composing the request (task generation) to be covered for a single satellite. A spot is assumed to be covered by a single strip.

RequestData:

- **tasks: Task[]** : the list of all submitted tasks.

A task structure derived from a request for information (RFI) along with its related attributes and requirements may be defined as follows:

Task:

- **id: string** : ID of this task, unique within RequestDB.tasks;
- **areaOfInterest: AreaOfInterest** : the AOI for this task;
- Type/goal (e.g. detect, locate, track, classify, identify, confirm, monitor, search, assess outcome, surveillance/survey, reconnaissance):
    - Mission tasks: survey, monitoring, tracking, search, detection, classification, identification, confirmation and outcome assessment.
- Complexity (task leaf nodes):
    - Primitive (mono); and

  - Compound (non-primitive/abstract/decomposable – e.g. stereo).

- `priority: double in [0.0, 1.0]:` Priority-nominal relative task value (value of information - information need –driven;

- Indicators, measure of performance (MOP) and measure of effectiveness (MOE) for goal achievement (e.g. quality of service (QoS), coverage, probability of detection, resolution, uncertainty, value of information (VoI) measure – e.g. National Imagery Interpretability Rating Scales (NIIRS)-like along with related thresholds);

- Collection requirements:

  - Imaging completion time window (mission/customer -driven);

  - NIIRS – NIIRS rating requirement:

    - Hint on proper resolution mode (beam/waveform).

  - Resource capability requirements;

  - Observation quality:

    - Observation angle; or

    - Acceptable incidence angle interval (min/max off nadir angle).

  - State conditions (e.g. environmental prerequisites, luminosity - sun orientation);

  - Direction (ascending/descending/none -orbits); and

  - Single/multiple possible observations per task (e.g. detection, confirmation), and periodicity - repeat cycle (e.g. once a day).

- Task tree/network (tree/directed acyclic graph – task structure decomposition):

  - Task dependencies and relationships (gathered information impacting/feeding other tasks) derived from the intelligence requirements management (IRM) process (assumed to be available – e.g. from CSIAPS);

  - Nominal value attached to an abstract IR parent node (request) is broken down among IR children nodes and propagated down to the task leaf nodes for which a collection tasking plan is required. Translating information needs, the basic/original intelligence collection tasking mission (request) derived from CCIRM commander's critical information requirements and the IRM process defines the root of the tree structure; and

  - The overall collection value (value of information) combines nominal value and quality of information/service associated with a task collection plan. It sums contributions over leaf IR node (task) collection values.

AreaOfInterest:

- **type: enum {SPOT, POLYGON_AREA}** : The type of this AOI. Spot targets are always fully covered by corresponding opportunities, while polygon area may require one or more opportunities for a complete coverage (may also not possibly be fully covered).
- Specification: vertex coordinates and centroid.

1.1.5 Set of collection assets (collectors)

CollectorData: collection assets. Space-borne platforms or spacecraft vehicles, namely Earth observation satellites:

- **platforms: Platform[]** : list of all available platforms; and
- **sensors: Sensor[]** : list of all known sensors from all platforms. Identical sensors on different platforms may or may not be duplicated in this list.

Platform:

- **id: string** : ID of this platform, unique within CollectorDB.platforms;
- **name: string** : human readable name (for simulation purposes, may be removed at a later time);
- Payload: single sensor;
- Range, altitude, speed, period, autonomy/endurance;
- **maxEnergyBudget: double** : budget per orbit typically in Joules. Maximum budget, actual budget on a given orbit (it may be lower);
- **maxObservationTime: double** : observation time per orbit in seconds. Maximum observation time as an estimation of thermal constraints. Actual value on a given orbit can be lower;
- **maxObservationCount: int** : maximum number of observations per orbit. Actual allowed number of observations on a given orbit can be lower;
- **maxMemoryStorage: double** : memory storage in MB (?). Maximum usable. Actual value on a given orbit can be lower; and
- Communication bandwidth.

Sensor:

- **id: string** : ID of this sensor, unique within CollectorDB.sensors;
- Types: optical (e.g. panchromatic, hyper/multi -spectral, infrared), AIS synthetic aperture radar – SAR;
- Phenomenology and modalities (e.g. acoustic, optical, infrared);

- Visibility (field of view/footprint), capability:
    - Range;
    - Mode number of bands:
        - Spectral and spatial resolution sensors (beam waveforms, mode, bands); and
        - Swath width, parameter configuration.
    - Min/max off nadir angle.
- Observation model (likelihood function, performance vs operational conditions):
    - Conditional probability distribution (probability of correct observation, false-alarm rate, given state/environmental conditions - clouds, weather forecast).

1.1.6 Sensor–task matchmaking

Determines feasible sensor package/resource capability and task requirements pairings. Typically achieved through semantic matchmaking, knowledge-based reasoning and/or classification analysis.

`MatchData:`

- **`opportunities: Opportunity[] :`** Sensor-task matchmaking assumed to be provided by CSIAPS.

Matching depends on many factors including:

- Cost;
- Asset capability:
    - sensor range effectiveness, technical characteristics, day/night operations reporting timeliness, geolocation accuracy, durability, sustainability, vulnerability.
- Environment task and requirements:
    - Operations conditions, task coverage, impact of adversarial planning, obstacles, weather, contingencies, kinematics, terrain obscuration, effective concealment, camouflage, and deception, risk, threat activity and behaviors;
    - NIIRS level or rating (imagery collection)/ resolution mode;
    - Duration, imaging time windows;
    - Sensor requirements: conditions:
        - Optical: weather, energy (sun visibility), target visibility and look angles; and
        - Beam waveform (resolution).

- Performance profile:
    - Feasible sensor-task performance model - sensor-task assignment quality/score;
    - Intelligence source performance profile – observation models; and
    - Collection probability of success.

`Opportunity:` feasible sensor-task pairings and measures of performance:

- **`id: string :`** ID of this opportunity, unique within MatchDB.opportunities;
- **`taskId: string:`** task ID in **`requests.tasks;`**
- **`platformId: string:`** platform ID in **`collectors.platforms;`**
- **`sensorId: string:`** sensor ID in **`collectors.sensors;`**
- **`startTime: double :`** in seconds, time of acquisition start from a given referential (e.g. 0 = now). Resource must have completed its transition/slewing before that time;
- **`endTime: double`** : in seconds, end time of acquisition;
- **`coverage: double in [0.0, 1.0] :`** relative area coverage;
- **`probability: double in [0.0, 1.0]:`** probability of success. *Always 1.0 in current implementation;*
- **`quality: double in [0.0, 1.0]:`** expected image quality. *Always 1.0 in current implementation*;
- **`cost: double`** : in $, imagery cost (for this whole image); and
- Opportunity and mutual conflicts.

1.1.7 Constraints set

Constraints set are defined along three separate dimensions, namely, task, resource and temporal. This separation is conceptual and not directly reflected in data models.

`ConstraintData:`

Task:

- Fairness between customer requests;
- **`maxTotalCost: double :`** maximum imaging acquisition cost overall;
- Overall energy budget;
- **`areaOverlaps: TaskAreaOverlap[] :`** for each opportunity pair (in MatchDB) for this task, this gives the overlap area, relative to the total AOI:

- Note: to reduce size, this array store pairs only once. For example, an overlap between opportunities “opp5” and “opp23” can be stored either as {“opp5”,”opp23”} or {“opp23”,”opp5”}.

- **`taskConstraints: string -> TaskConstraint`** : per-task constraints (one per task). Map key is the task ID.

Resource:

- Asset inventory;
- Task visibility (per orbit);
- **`revolutionPeriod: double :`** in seconds, revolution period for all satellites and all orbits, currently assumed to be the same. Will eventually be explicit for each platform (and perhaps each orbit);
- **`platformCapacities: string -> PlatformCapacity :`** satellite platform capacity (one per satellite). Map key is the platform ID;
- Utilization/deployment:
  - Sensor interference or sensor mixes contingencies;
  - Environment: weather conditions:
    - Cloud cover. Some sensors cannot see through clouds. Not only do clouds cover much of the Earth at any given time, but some locations are nearly always cloudy.
  - Routine operations:
    - periodic maintenance, recharging batteries, on-board data processing, orbit corrections, uplink.

    Conflicting opportunity over single platform utilization due to contention between observation opportunities (e.g. mutual exclusion of competing concurrent sensing actions over waveforms/beam modes).

Temporal:

- Set-up time: start-up, shutdown, recovery, altitude stability, processing, replanning;
- **`transitions: OpportunityTransition[]:`**
  - Note: transition time, slewing (rates)/moving duration to switch between consecutive observation angles from a predecessor to a successor opportunity task. Transition between look angles can be achieved through instruments mounted on motors that can point either side-to-side (cross-track), forward and backward along the track, or rotate to point their instruments in any direction (agile satellites);
  - This array only contains **`OpportunityTransition`** instances from an opportunity Oi an opportunity Oj where:
    - Oi and Oj occur on the same platform;

        - Oj starts after Oi; and
        - Oj occurs no more than one revolution period after Oi (beyond that time gap, transition information is deemed irrelevant).
    - If a transition from Oi to Oj is not found, it can be assumed that transition duration is 0.0 (either because it does not apply or is deemed irrelevant).

- Time windows/intervals: availability/serviceability:
    - Service – task imaging (customer requirement);
    - Conflicting opportunities (time window overlaps);
    - Dissemination –images downlink (image delivery to customers for consumption);
    - Visibility – task opportunity visibility for imaging (agile satellite); and
    - Downlink to specific ground station network nodes.
- Information age – maximum delay between observation and download;
- **`timeHorizon: double`** : in seconds, scheduling time horizon (e.g. 24 hours * 3600).

`TaskAreaOverlap:` area overlap between two opportunities on the same task:

- **`opportunityId1: string :`** first opportunity ID in **`MatchDB.opportunities;`**
- **`opportunityId2: string :`** second opportunity ID in **`MatchDB.opportunities;`**
- **`area: double :`** area overlap relative to the total AOI. For example, for two opportunities with relative coverage 0.5 and 0.8, overlap would be within 0.3 to 0.5 (i.e. not greater than any of those two, and large enough so that arithmetical sum of those areas less the overlap is <= 1.0);
- **`quality: double :`** compound quality of both opportunities. *Always 1.0 in current implementation*.

`TaskConstraint:` various constraints that apply to a specific task:

- Energy budget;
- Basic task collection requirements;
- Task precedence (e.g. survey, detect, track, identify and confirm sequence);
- Single (successful observation assumption - probability of success = 1) or multiple observations/visits per task (e.g. detection, confirmation);
- Preemptive or not;
- **`maxCost: double :`** allowed budget;
- **`minCoverage: double in [0.0, 1.0]:`** area coverage threshold (relative to original AOI); and

- Probability of detection/ entropy thresholds (quality of information/service task requirements).

`PlatformCapacity:`

- Revolution period $T\rho$ for each orbit (currently in **`ConstraintDB.revolutionPeriod`**);
- Payload;
- Communication (uplink, downlink): bandwidth, communication cost rate, downlink energy consumption;
- UAV endurance; and
- **`orbits: PlatformOrbitCapacity[] :`** per orbit data (one **`PlatformOrbitCapacity`** instance for each orbit ρ of the platform).

`PlatformOrbitCapacity:` data for a given orbit ρ of a platform:

- **`startTime: double`** : in seconds, the orbit start time for that platform. Time referential is not specified but it is the same for all platforms in a **`DataSet`**;
- **`endTime: double`** : in seconds, the orbit end time for that platform. It is excluded from the current orbit and typically the same as the next orbit start time. Time referential is not specified but it is the same for all platforms in a **`DataSet`**;
- **`memoryStorage: double :`** in MB, $W\rho$: memory storage capacity in orbit ρ. limited on-board data storage. Images are stored in a solid-state recorder (SSR) until they can be sent to the ground;
- **`imagingMemoryRate: double :`** in MB/s, $w\rho$ memory consumption rate by observation time in orbit ρ.;
- **`energyBudget: double`** : in Joules, $E\rho$: energy capacity in orbit ρ.;
- **`imagingEnergyRate: double`** : in J/s, $eo\rho$: energy consumption rate by observation time in orbit ρ.;
- **`transitionEnergyRate: double`** : in J/s, $es\rho$ energy consumption rate for task *transition* time (see **`OpportunityTransition`**) by a sensor in orbit ρ. This does not apply to the platform/sensor-specific *set-up* time, only *transition* time;
- **`maxObservationCount: int :`** $c\rho$: maximum number of sensor openings in orbit ρ.;
- **`maxObservationDuration: double :`** thermal: maximum satellite imaging time per orbit; and
- **`setupDuration: double :`** base set-up time for any observation. Can be 0.0 on some platform/orbit. Does not include any transition time (must be computed separately on an opportunity-to-opportunity basis).

`OpportunityTransition:` data pertaining to the transition from an opportunity to another, occurring on the same platform:

- **`fromId: string`** : the ID of the first opportunity in **`MatchDB.opportunities`**;

- **`toId: string`** : the ID of the next opportunity in **`MatchDB.opportunities`**; and
- **`duration: double`** : in seconds, required *transition* time (slewing and/or other delays) to be ready for second opportunity. Does NOT include platform/sensor-specific *set-up* time.

1.1.8 Collection tasking objective

The collection tasking objective (open-loop multi-satellite collection scheduling) has no data in the current implementation. As such, it is not part of the provided data set. It is, however, provided through a set of functions, for the currently implemented part.

`ObjectiveFunction:` no data yet, only code:

- **`Single objective`**: max expected value of information collection task. Not provided as data in current implementation but rather through DISCOVER objective function upper bound (see section 2 Objective Function);
- Multiple objectives: weighted sum:
  - Relative weights (sum = 1);
  - Expected task value (over [0,1]);
  - Imaging cost ($) (table lookup – from CSIAPS);
  - Demand satisfaction (number of serviced tasks); and
  - Lateness, resource and energy consumption, risk exposure, make span.

1.1.9 Pre-Defined Task Plans

The data set allows to specify pre-defined plans for some tasks. There can be more than one possible plan for any given task.

Those plans may come either from external data sources or from different solver. For instance, QUEST can handle spot targets but may only handle tasks with large polygon AOI if there are pre-defined plans for those (i.e. it may only choose one item to cover a task, either an opportunity or a pre-defined plan). On the contrary MOSAIC is designed to generate plans for a single large area task. Therefore, in a two-step process, MOSAIC could generate plans for non-spot tasks, add them to the data set, and feed that to QUEST.

MY-PICC and GATHER do not support pre-defined plans (they simply ignore them) and only QUEST is meant to use them. This is not yet fully implemented but should be in a future revision. MOSAIC is currently the only potential source (yet to be fully implemented too), as none of the simulation data (as described in section 6) provides pre-defined plans. Therefore, current implementation uses a simplistic approach, where pre-defined plans are a map from task IDs to a list of **`TaskPlan`** for each of them.

`TaskPlan:` simple task plan:

- **`taskId: string`** : the ID of the related task; and
- **`opportunities: Opportunity[]`** : the list of all opportunities in the task plan. Those are direct in-memory references to opportunities in the current data set.

1.1.10 Not implemented yet

- *Sppt*: set of supporting resources – typically, ground stations:
    - Image downlink scheduling is ensured through a ground station network. Ground station and relay node attributes:
        - Locations;
        - Mask (maximum communication range); and
        - Communication bandwidth and channels.
- *A*: Action space/set:
    - Sensing actions/controls:
        - Mode (waveforms/beams and bands), polarization, and other control actions; and
        - Sensor orientation (camera/antenna pointing angles along some rotation axes – roll pitch, yaw).

1.2 Collection Plan (CP) Data Model (1st draft)

There are no precise usage/functional requirements for the collection plan so far, although some are obviously expected. As such, a collection plan is merely defined as a solver output for now, and further design will apply along with forthcoming requirements.

In detail:

- All CP-related Java classes are packaged along with data model classes, under the `DiscoverCommons` module:
    - See `ca.gc.rddc.discover.commons.dataaccess`.
- `CollectionPlan` is the main class; and
- Current implementation is trivial

`CollectionPlan:` solver result:

- `taskPlans: String -> TaskPlan` : the unique task plan for each covered task (i.e. with at least one selected opportunity).

2 OBJECTIVE FUNCTION

This function implements an evaluation of a collection plan with regards to the collection tasking objective (open-loop multi-satellite collection scheduling). It represents the "value of information" for a given collection plan, which solvers aim to maximize. It is used directly by all solvers, except for CPLEX (IBM ILOG CPLEX Optimizer, a software MIP solver package)-based solvers. The latter rather rewrites it as sets of linear constraints.

2.1 Mathematical model

2.1.1 Data

Functions use the following data variables:

- P : set of all satellites orbits/passes ρ:
    - Sorted in chronological order. Whether they are sorted by satellite then time, or by time then satellite, is implementation-defined and does not affect the model, as long as they are at least chronologically sorted for any given satellite.
- R : set of candidate requests *r*, each defining a target (spot) or polygon area (region) of interest (AOI) to be covered:
    - Each request *r* is decomposed in a set of tasks;
    - A compound request includes more than one task; and
    - A spot request generally corresponds to a single primitive request involving one task. However, some primitive request/task requirement might require more than one task, e.g. a stereo task necessitating an Ascending and a Descending imaging collection.
- S_r : set of tasks/subtasks *s* composing request *r*:
    - In current implementation, task and requests are assimilated, as if there was only one task per request, and solvers deal directly with tasks as if they were requests.
- V_{rs0} : normalized nominal value of request-task (*r, s*):
    - Within [0, 1]
- $O_{rs\rho}$: set of imaging opportunities *o*, for request-task (*r, s*) during orbit ρ:
    - Sorted chronologically on their imaging start time.
- A_{rs} : AOI associated with request-task (*r, s*):
    - In current implementation, it is not described in geographic terms. It is only assumed to exist and as such is bound to the constant 1 in formulas below.
- $A_{rso\rho}$ (= A_o): partial area coverage of A_{rs} , resulting from intersection of A_{rs} opportunity *o* strip area on orbit ρ:

- In current implementation, such partial coverage is not described in geographic terms. It is rather provided as a relative coverage to A_{rs}, within [0, 1].

- $A_{rso\rho o'\rho'}$ (= $A_{oo'}$): partial area coverage of A_{rs}, resulting from the intersection of $A_{rso\rho}$ and $A_{rso'\rho'}$:
  - As with $A_{rso\rho}$, provided as a relative coverage to A_{rs}, within [0, 1].

- $p_{rso\rho}$ (= p_o): probability of imaging success associated with opportunity *o* on orbit ρ for request-task (*r, s*):
  - Within [0, 1]

- $q_{rso\rho}$ (= q_o): estimated imaging opportunity normalized quality associated with opportunity *o* on orbit ρ for request-task (*r, s*):
  - Within [0, 1]

- $q_{rso\rho o'\rho'}$ (= $q_{oo'}$): composed imaging opportunity quality associated with intersecting opportunity *o* and *o'* strip areas related to orbit ρ and ρ' respectively, for request-task (*r, s*):
  - Within [0, 1]

- $x_{rso\rho}$ (= x_o): binary variable indicating whether opportunity *o* is selected (1) or not (0) in a given collection plan. That opportunity services request-task (*r, s*) in orbit ρ.

- *Plan*: the set of all $x_{rso\rho}$ over P, R, all S_r and all $O_{rs\rho}$.

2.1.2 Formulas

Ultimately, solvers aim to maximize the value of Obj_{Plan}, the objective value of a full collection plan. But in the process, they also need parts of it for localized or incremental calculations. The objective function is thus decomposed into the following functions:

Plan objective value:

- $Obj_{Plan} = \max\left(\sum_{r\in R}\sum_{s\in S_r} Obj_{rs}, 0\right)$

- Lower bound: $\inf Obj_{Plan} = 0$

- Upper bound: $\sup Obj_{Plan} = \sum_{r\in R}\sum_{s\in S_r} \sup Obj_{rs}$

Request-Task objective value:

- $$Obj_{rs} = V_{rs0} \sum_{\rho \in \mathrm{P}_{rs}} \sum_{o \in O_{rs\rho}} \left(\frac{A_o}{A_{rs}} p_o q_o x_o - \sum_{\rho' \in \mathrm{P}_{rs}} \sum_{\substack{o' \in O_{rs\rho'} \\ o<o'}} \frac{A_{oo'}}{A_{rs}} p_o p_{o'} (q_o + q_{o'} - q_{oo'}) x_o x_{o'} \right)$$

- Where $o < o'$ ensures that opportunity intersections are counted only once over all opportunities for the given request-task;

- Lower bound: unbounded.

    - Result can be negative if for one or many opportunities, the sum of contributions from intersections (second term) is greater than the opportunity own contribution (first term).

- Upper bound: $\sup Obj_{rs} = V_{rs0} \min\left(1, \sum_{\rho \in \mathrm{P}_{rs}} \sum_{o \in O_{rs\rho}} \frac{A_o}{A_{rs}} p_o q_o \right)$

    - This is a conservative theoretical upper bound, where all associated opportunities are selected;

    - It assumes only positive opportunity contributions, without subtracting intersections; and

    - It limits the sum to 1, as if the request-task AOI was fully covered with maximum quality and probability.

Opportunity differential objective value:

- Solvers often have to compare opportunities to add/remove them to/from a plan while building the latter. The differential objective value gives the variation of the associated request-task objective value when an opportunity is added/removed;

- $$DiffObj_{rso\rho} = \frac{A_o}{A_{rs}} p_o q_o x_o - \sum_{\rho' \in \mathrm{P}_{rs}} \sum_{\substack{o' \in O_{rs\rho'} \\ o \neq o'}} \frac{A_{oo'}}{A_{rs}} p_o p_{o'} (q_o + q_{o'} - q_{oo'}) x_o x_{o'}$$

- Lower bound: unbounded, as with request-task objective value;

- Upper bound: 1; and

- While this looks similar to Obj_{rs} summed terms Obj_{rs} is *not* directly the sum of all associated $DiffObj_{rso\rho}$. Indeed, in such sum, opportunity intersections would be counted twice, as the above only restricts o' with $o \neq o'$ instead of $o < o'$. However, in an iterative process where opportunities are added/removed (i.e. where x_o are set/reset) one by one, Obj_{rs} *can* effectively be computed with the sum of all $DiffObj_{rso\rho}$ .with varying x_o on each iteration.

Single opportunity value:

- Merely the value of an opportunity, as if it was alone as part of a task (no overlap);
- $Obj_{rso\rho} = Obj_o = \frac{A_o}{A_{rs}} p_o q_o x_o$
- Bounded to [0, 1].

Request-Task relative coverage:

- Provides the relative coverage of a request-task AOI by a set of opportunities associated to that request-task;
- $$RelCov_{rs} = \sum_{\rho \in \mathrm{P}_{rs}} \sum_{o \in O_{rs\rho}} \left(\frac{A_o}{A_{rs}} x_o - \sum_{\rho' \in \mathrm{P}'_{rs}} \sum_{\substack{o' \in O_{rs\rho'} \\ o<o'}} \frac{A_{oo'}}{A_{rs}} x_o x_{o'} \right)$$
- This is derived from the request-task objective value, but it considers only the covered area, no matter the quality and assuming it is effectively covered (as if $p_o = q_o = 1$);
- Lower bound: unbounded. Although counter-intuitive, coverage can be negative, as with request-task objective value and for the same numerical reasons.
    - This is a numerical artifact because above formula does not compute actual union of coverage from all opportunities. Indeed, it only considers first order intersections (pairwise) but not higher-order intersections (three-by-three, four-by-four, etc.); and
    - A real-world description of AOIs (such as geographic coordinates) could fix this issue, although the request-task objective value would have to be computed accordingly as well, for consistency.
- Upper bound: 1.

2.1.3 Wang objective function variant

For the sake of comparison among solvers and with external sources, a Wang-specific objective function is used to work with Wang data sets (see section 6.2 Wang data sets). Wang data sets don't have relative coverage (only spot targets, task coverage is assumed to be 1.0 on all opportunities) nor quality in their opportunities (assumed to be 1.0 as well).

Therefore, the Wang objective function is similar to the above DISCOVER objective function but only considers task nominal values and opportunity probabilities. This nevertheless allows for some refinement as a proper task value can be computed for more than two opportunities. In comparison, DISCOVER task value calculation is otherwise limited to an estimation, because overlaps between opportunities are limited to pairwise intersections in the data set model. The formulas, below, become rather simple:

Plan objective value:

- $Obj_{Plan} = \sum_{r \in R} \sum_{s \in S_r} Obj_{rs}$
- Lower bound: $\inf Obj_{Plan} = 0$
- Upper bound: $\sup Obj_{Plan} = \sum_{r \in R} \sum_{s \in S_r} \sup Obj_{rs}$

Request-Task objective value:

- $Obj_{rs} = V_{rs0}\left(1 - \prod_{\rho \in \mathrm{P}_{rs}} \prod_{o \in O_{rs\rho}} (1 - p_o) x_o\right)$
- Lower bound: $\inf Obj_{rs} = 0$
- Upper bound: $\sup Obj_{rs} = V_{rs0}\left(1 - \prod_{\rho \in \mathrm{P}_{rs}} \prod_{o \in O_{rs\rho}} (1 - p_o)\right)$
    - Just like request-task objective value, but assuming all opportunities are selected ($x_o = 1$ for all opportunities).

Opportunity differential objective value:

- $DiffObj_{rso\rho} = 1 - (1 - p_o) \prod_{\rho' \in \mathrm{P}_{rs}} \prod_{\substack{o' \in O_{rs\rho'} \\ o \neq o'}} (1 - p_{o'}) x_{o'}$
- Bounded to [0, 1]

Single opportunity value

- $Obj_{rso\rho} = Obj_o = p_o x_o$
- Bounded to [0, 1]

Request-Task relative coverage

- Trivial: it is 1 if at least one opportunity is selected for that task, 0 otherwise.

2.2 Java Implementation

Package `ca.gc.rddc.discover.commons.dataaccess` contains the interface `ObjectiveFunction` which provides the definition of above functions.

Implementation of those functions, as described above, is found in package `ca.gc.rddc.discover.solvers.common` in `DiscoverObjectiveFunction` class (and `WangObjectiveFunction` for the Wang variant). It is a straightforward formula translation to Java code, except for the binary $x_{rso\rho}$ variable. Java methods are rather defined to accept the set of selected opportunities themselves as input parameters.

Separating the interface from the implementation allows for solvers to be reused with different objective functions, as there could be in the future, as long as they implement the same interface.

2.3 Test and Validation

Given the fairly trivial nature of such functionalities, a thorough coverage via unit tests is enough to assess code validity.

3 CPLEX-BASED MODELS/SOLVERS

This section focuses on an exact algorithm solver in DISCOVER named QUEST (QUadratically constrained cplEx-based Solver (Scheduler) Technology) and a more advanced version of this model, RE-CVEST (delayed REward Collection Value cplEx-based Scheduler Technology). RE-CVEST is only a model improvement of the original model of QUEST so that it can solve very large size and complex problems. This section also discusses a sub-module of QUEST called MOSAIC (Mixed/multi Overlapping Sub-Area composition Coverage) solver which finds feasible plan for large areas. All those algorithms aim to solve parts of the Multi-Satellite Collection Scheduling Problem (MSICSP), as stated in the *Statement of Work.*

3.1 Summary

In short, the MSICSP integrates three well-studied problems, namely determining tasks to cover (coverage problem), using what resources (assignment problem) and when (scheduling problem), all of that while respecting limited resource constraints. Given a set of tasks to service over a multi-period horizon, a MSICSP solver aims at finding a feasible combination of imaging opportunities such that total collection value is maximized.

As a reminder, each image acquisition request has associated AOI coverage plan(s). Each AOI coverage plan consists of strip(s). The strips may be overlapping. Each strip has one or more opportunities to be imaged during scheduling horizon [0, T]. Each opportunity of a strip allows the acquisition of the entire strip. Each opportunity has certain properties inherited from requirements of the request.

An imaging opportunity is thus associated to a task request and the geographic fraction of its AOI, a satellite and sensor (resolution/mode) that performs it, a time window and a probability of success (presence of clouds for instance).

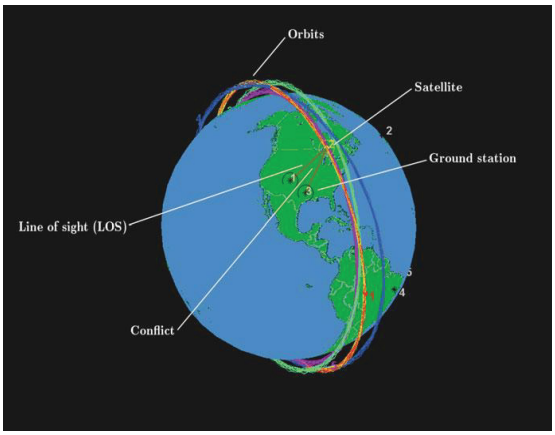


Figure 1: Scheduling scenario

Furthermore, the MSICSP involves operational constraints such as maximum energy available, maximum memory storage, thermal constraint, total budget available, budget per task and maximum number of opening constraints.

In the proposed algorithm, in order to improve the possibility of completing tasks given variable opportunities probability, we account for the scheduling of each task over multiple resources and establish a novel non-linear mathematical model. The novelty of this approach lies in exploring the mathematical structure to have both linear (Coarse-Grained) and quadratic (Fine-Grained) models. The Coarse-Grained model deals with the user default plans while the Fine-Grained model solves the problem without any default plans. QUEST explores these two mixed mathematical models (Coarse-Grained and Fine-Grained) to tackle this problem. Also, QUEST can compute a very tight upper bound of the problem which is very useful for any other approach or algorithm in order to know solution quality.

This problem can be modeled as a multi-network-flow problem with resource constraints, a well-known model which is NP-Hard combinatorial optimization problems. They have been successfully used in structured Vehicle Routing and combinatorial optimization problems, involving ground vehicles. The specific objective of the QUEST project is to demonstrate that exact algorithms can as well be applied to these problems involving multiple aerial multi-satellite image scheduling.

The algorithm is fully implemented as part of the QUEST solver, and experiments were conducted to validate feasibility, robustness, and efficiency of this approach. QUEST-specific code is available under:

- `DiscoverSolvers` module; and
- `ca.gc.rddc.discover.solvers.mypicc` package.

We show that our Zero Half Cuts combined with branch-price-and-cut algorithm clearly outperform a state-of-the-art branch-and-cut algorithm on the hard instances. Experimental results based on the Wang Data and Berger Data test generator (harder instances) show that the solutions of our models perform better than those of previous studies and provide very good upper-bound of the original problem, and we also reveal the strengths and weaknesses of the proposed algorithms while solving different size and complexity instances. Finally, a series of test examples and comparisons are given out, which demonstrate clearly that our algorithm performs better than five heuristics approaches. The potential collection value gains range from 15 to 30% compared to the best heuristic solution.

Project Goal: Develop collection management decision support capabilities for effective and efficient satellite resource utilization and enhanced intelligence collection capabilities, through the development of relevant concepts, models, and algorithms.

3.2 Problem Description

The problem addressed by QUEST is a very difficult scheduling problem. First, the original objective function is not linear. Second, the operational constraints are resource constraints (total budget to perform all tasks, thermal budget, maximum opening, energy budget, memory budget and maximum budget per task). Finally, energy constraint is one of the most difficult, because it involves all variable arcs of the acyclic graph of an orbit (too many arcs in some instances of acyclic-graph). In this project, we exploit the mathematical structure of this problem in a non-obvious way in order to make it linear and/or quadratic so that to be able to use the power of existing well-known exact algorithms to tackle this problem.

Based on a non-deterministic setting (uncertainty of cloud) in which image acquisition is characterized by a probability of successful observation, the single episode (static) multi-satellite collection scheduling/tasking can be stated as follows: given a set of information requests translating task areas of interest to be observed (weighted/valued tasks), a set of non-agile heterogeneous satellites (collection assets), and a set of imaging opportunities, the problem consists in allocating collection assets to imaging observation task opportunities in order to maximize expected collection value or profit over all requests and a specific time horizon, subject to a set of constraints (e.g. mission, task, operational, collector, supporting resource, communication, capacity, temporal and cost).

3.3 Motivation and Context

Multi-mission (satellite) collection scheduling with automated and integrated utilization of heterogeneous satellite sensors would result in the increased performance of intelligence collection beneficial to a variety of operations including:

- Maritime surveillance and ship detection;
- Arctic land surveillance;
- Environmental sensing; and
- Search and rescue.

There are many other applications:

- *Satellite communications*, where tasks are communication intervals between sets of satellites and ground stations;
- *Earth observation*, where tasks are observations of spots on the Earth by satellites; and
- *Sensor scheduling*, where tasks are observations of satellites by sensors on the Earth.

Examples:

- ESA's Tracking Station Network (ESTRACK) had 30 satellites and 40 antennas;
- NASA's Deep Space Network (DSN) had 35 satellites and 13 antennas (with 425 requests per week); and
- Air Force Satellite Control Network (AFSCN) had 100 satellites and 16 antennas (with 500 requests per day).

The design and development of the centralized system with the tools for generating optimized multi-mission collection schedules would allow the integrated utilization of Canadian satellite resources and it would have the potential of increasing the overall operational performance by 25-40% ([2]and [3]). These potential gains are confirmed by this study for big and hard problem instances.

3.4 Literature Survey

The following table includes this work along with previous work.

Table 1: Summary of the Literature on Image Acquisition Scheduling for Multi-Satellite Collection Scheduling

Authors	Year	Satellite(s)	Objective	Method	Problem size	Solution Quality
Exact Algorithms						
Hall and Magazine [4]	1994	1S		**Dynamic programming** algorithm with a bounding by either Lagrangian relaxation or relaxation of some constraints	200 targets	
Gabrel and Vanderpooten [5]	2002	1S, 4O, Non-agile	Maximize the number of high priority shots; Maximize demand satisfaction; and Minimize satellite use to prevent wear.	Model: multi-criteria path problems over an acyclic graph. Algorithm: **the label-setting shortest path algorithm** for generating all efficient paths, followed by an interactive session.		Less than an hour running time
This work	2017	3S, 21O, Non-Agile	Maximize the number of high priority and quality shots; and Maximize demand satisfaction.	Model: Coarse-grained and Fine-grained over an acyclic graph. Algorithm: MIPs and QMIP	160, 200 tasks	Less than one minute
This work	2017	3S, 42O, Non-Agile	Maximize the number of high priority and quality shots; and Maximize demand satisfaction.	Model: Coarse-grained and Fine-grained over an acyclic graph. Algorithm: MIPs and QMIP	160, 200 tasks	Less than one minute
This work	2017	3S, 15O, Non-Agile	Maximize the number of high priority and quality shots; and Maximize demand satisfaction	Model: Delayed rewards over an acyclic graph. Algorithm: MIPs with strategy to find good integer solution.	300 tasks	Less than 2 minutes
Greedy						
Muraoka et al.	1998	1S, ASTER	Ranking function consists of 12 elements.	**Greedy**: one by one request is scheduled.		

Authors	Year	Satellite(s)	Objective	Method	Problem size	Solution Quality
Heuristics						
Pemberton	2000	1S		Priority **segmentation**: each sub-problem solved to optimality by **branch and bound**.		**Priority segmentation is** better than greedy algorithm that sorts tasks by priority and schedules them one by one.
Frank et al.	2001	1S	Ranking: priority and contention (how many opportunities a request has).	**Greedy** hill-climbing search with **stochastic** variations to escape local optima. Algorithm based on **constraint-based interval planning.**		
Metaheuristics						
Bensana et al. [6]	1996	1S, 1O or mO, SPOT 5 with 3 cameras. Mono-image and stereo-image requests. Non-agile		Depth-First **Branch and Bound**, **Russian Doll Search**, Best-First Branch and Bound, a **Greedy** Algorithm, and **Tabu** Search. Integer Linear Programming (ILP) and Valued Constraint Satisfaction Problem (VCSP) formulations.	13 instances 1S1O, and 7 instances 1SmO	**Tabu Search** performed best in general, Russian Doll Search performed best for the one orbit case, but failed in six of the seven instances in the multiple orbit cases.
Gabrel et al. [7]	1997	1S, Non-agile		An **approximate** method with discrete times, an approximate method with cont. time and **greedy** algorithm based on the **longest path** algorithm, and an exact method **branch and bound** approach with a depth-first search strategy		The **continuous-time approximation** algorithm averaged within 7% of optimal. The discretized time algorithm performed very close to the continuous approximate one.
Wolfe and Sorensen [8]	2000	1S and 2S		**Hill-climbing**, Look-ahead algorithm, **Genetic** algorithm with look ahead. Defined the problem as a window-constrained packing problem.	Time horizon: One week	**Genetic** algorithm performed the best. Needed longer run times. Look-ahead algorithm may be more practical for large problem instances.
Vasquez and Hao, 2001 [9]	2001	1S, Non-agile		**Tabu** search together with “logic-constrained” knapsack formulation.		Better than tabu search of [6]

Authors	Year	Satellite(s)	Objective	Method	Problem size	Solution Quality
Lemaitre et al. [10]	2002	1S, 1O, with stereo-images. **Agile** satellite		**Greedy** algorithm, **dynamic programming** algorithm, **constraint programming**, and **local search** algorithm. Greedy and dynamic algorithm ignored stereo images.	6 problem instances	Only mono-images: dynamic programming approach was the best performer; Stereo-images included: **local search** methods were the best.
Globus et al. [2]	2003	1S and 2S, **Agile** Optical, included (as for SAR): access, slew, and dwell times. Considered: on-board data storage and downlink	Minimize: - Weighted sum of unscheduled image priorities; - Total slew time; and - Sum of slew angles of scheduled images.	**Genetic** algorithm, **Simulated** annealing, **Stochastic hill-climbing**, **Iterated sampling.**	4200 imaging targets with priorities 1 to 5	**Simulated** annealing was the best.
Globus et al. [11]	2004	2S and 3S, Optical satellites. Same as Globus et al., 2003 [2]	Minimize: - Weighted sum of unscheduled image priorities; - Mean slew time; and - Mean of off-nadir pointing angles of scheduled images.	**Genetic** algorithm (13 variations), **Simulated** annealing, Stochastic hill-climbing, **Squeaky wheel**.	10 instances, 2100*m targets with priorities, m - number of satellites	**Simulated** annealing was the best, closely followed by hill climbing. Genetic algorithm was the worst.
Cordeau and Laporte	2005	1S,1O **Agile** satellite	Piecewise-linear convex with respect to the proportion of the polygon's area being acquired	**Tabu** search. Allowed infeasibility by relaxing the time window constraints. Upper bound: column generation procedure solving the linear program relaxation.	ROADEF 2003 competition	**Tabu search** achieved **near-optimal** solutions. Compared to upper bound.
Lin et al. [12]	2005	1S, China/ Taiwan ROCSAT II (FORMOSAT II)		**Lagrangian relaxation** and **linear search**		**Lagrangian relaxation** is better when compared to simple tabu search.
Lin and Chang [13]	2005	1S		Hybrid approach: **Lagrangian relaxation** with a **tabu** search based feasibility adjustment.		**Hybrid** better than Lin et al., 2005 [12]

Authors	Year	Satellite(s)	Objective	Method	Problem size	Solution Quality
Bianchessi et al. [14]	2007	2S, PLEIADES (French), Optical, **Agile.** Priority, consecutiveness of strips from one polygon request	Maximize weighted function of utilities assigned to users; linear with respect to the proportion of the polygon's area being acquired	**Tabu** search partly based on [Cordeau and Laporte, 2005].	Time horizon: 24 hours	The same as [Cordeau and Laporte, 2005]
Wang et al. [15]	2009	3S, mO		**Tabu** search heuristics embedded in ILOG Dispatcher and a **greedy**-based algorithm called "conflict-avoided heuristic".		**Tabu** search resulted in better solutions. Running time for their largest instance, the greedy took 50 seconds, while tabu search took 3499 seconds.
Wang et al.	2010	mS FORMOSAT 5, Taiwan	Multi-criteria	**Genetic** algorithm. Problem simplified by a division in limited numbers of single-orbit scheduling problems.	20 targets	Not provided
Nelson [3]	2012	2S, 2O, SAR satellite constellations: Walker, Target priorities		Three-step approach: cluster-route-schedule. Clustering groups imaging targets. Routing step uses **column generation** – in sub-problems each satellite is separately. Scheduling step for each satellite separately uses **time-space networks** and heuristics.	8,250 targets 4,271 could be imaged within 208 min time horizon (2 revolutions)	768 clusters formed
Wu et al. [16]	2013	4S, mO, 1day sch. hor.; Semi-agile (only roll) Spot target tasks. Priorities.	Maximize profit	Adaptive **simulated annealing** with dynamic clustering. Compared to ant colony, tabu search, genetic algorithm, simulated annealing with static clustering, highest priority first alg.	100-1000 targets, 1d	Adaptive **simulated annealing** with dynamic clustering.

Authors	Year	Satellite(s)	Objective	Method	Problem size	Solution Quality
Xiaolu et al.	2014	mS, mO, Semi-agile (only roll) Energy and memory are considered. Max continuous imaging time is considered.	Maximize profit	Decomposition algorithm: one sub-problem is the assignment of a task to an opportunity and the other sub-problem is the merging of tasks into an observation. The problems are solved iteratively until stopping criteria is met. MIP is proposed for scheduling observations for collection of satellites.	Random instances: uniform, collective, and mixed. 63 inst: 100-600 req, 2-8 sat -> 200-4300 opp	The proposed **decomposition** algorithm is better than simulated annealing: around 25% better and up to 50% for some instances. Operational in China Satellite Management Center.
Tangpattanakul et al.	2015	Agile. Includes stereo requests and polygon requests	**Multi-criteria**: max profit + max fairness (by minimizing the maximal difference between the user profits)	Iterated indicator-based multi-objective local search.	Modified ROADEF 2003	
Wang et al. [1]	2015	mS, mO, Semi-agile (only roll) Uncertainty of clouds	Maximize expected profit	Exact: Decomposition with master problem solved by enumeration and sub-problems by dynamic programming. Heuristics: for generating feasible solution to sub-problems. Master still solved by enumeration.	Three satellites. Planning horizon: from 6h to 24h. Number of requests from 10 to 160.	Compares to Liao et al., 2007 [17] model solved by CPLEX. Generates better solutions.
Malladi et al.	2015	mS, mO, Semi-agile (only roll)	Maximize priority	Model-based meta-heuristic (matheuristic). MIP. Also, modeling as a new cluster-restricted maximum weight clique problem.		Compares to CPLEX. Real-world problems instances. Also BOSHLIB and DIMACS benchmarks.
Constraint Programming (CP)						
Verfaillie et al. [18]	1996	1S, 1O, Non-agile		**Russian Doll algorithm** to improve search for an opt solution in CP; Depth-first search branch and bound.	8 instances, 30 min time limit	**Russian Doll** algorithm solved all instances to optimality, while branch and bound did not solve any.
Lemaitre et al. [19]	2000	1S, **Agile** satellite		**Constraint programming** (OPL Studio Framework), Local search (LS)	7 instances; Stop criteria: CP: 5 min, LS: 2 min	**Local search** algorithm was better than CP.

Authors	Year	Satellite(s)	Objective	Method	Problem size	Solution Quality
Bounds						
Gabrel and Murat [20] [21]	2003, 2006	1S, 1O, SPOT 5, 3 cameras, On-board memory, Non-Agile		**Vertex-path** formulation with a **column generation** procedure.	Up to 300 targets	Bounds generally within 10% of optimality compared to Bensana et al., 1996 [6]. 2006: show that the bounds are tighter than linear relaxation.
Vasquez and Hao [22]	2003	1S, mO, SPOT 5, Non-agile		Using **tabu** search the problem is partitioned into sub-problems, which are solved exactly by an iterative enumeration algorithm.		Equal or better than Gabrel and Murat, 2003 [20]. Upper bounds less than 3% from lower bounds.
Benoist and Rottembourg [23]	2004	**Agile** satellites		**Prize Collecting Traveling Salesman Problem with Time Windows**, enhanced with valid inequalities based on task interval reasoning and a Russian Doll Search approach.	20 instances with 300-500 targets	With valid inequalities: 22% gap. With Russian Doll search: 12% gap, took several hours.
Bianchessi et al. [14]	2007	2S, PLEIADES (French), Optical, **Agile.** Priority, consecutive-ness of strips from one polygon request	Maximize weighted function of utilities assigned to users; linear with respect to the proportion of the polygon's area being acquired.	**Column generation** based on set partitioning. Branch and price.	Time horizon: 24 hours	The same as [Cordeau and Laporte, 2005]

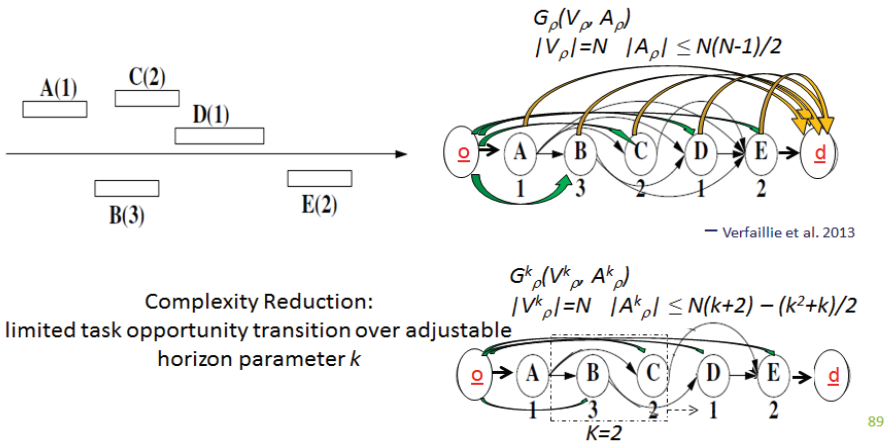
3.5 Graph and Orbit Description

Graph (orbit network – acyclic directed graph) reduction consideration: consider opportunity node (*r,s,o*) transition over a limited period only (state/task transition ending beyond that period is assumed unlikely to occur and will be ignored). For instance, a finite number of *k* immediate successors might be considered.

Source (*b*) and Sink (*e*) nodes in $G_\rho(V_\rho, A_\rho)$: $x_{b11\rho} = 1$, $x_{e11\rho} = 1$

V_ρ : set of opportunities {(*r,s,o*)} in orbit ρ ($V_\rho = O_\rho = \cup_{rs} O_{rs\rho}$)

A_ρ : set of arcs connecting node (*r,s,o*) to node (*r',s',o'*), reflecting feasible transition (between opportunities) in orbit ρ.

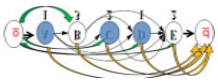


Graphs are built separately for each platform:

Platform 1

Platform 2

Platform 3



….

Mathematic model can be found in [24].

3.6 Original problem

3.6.1 Notation

H: time horizon

SAT: set of heterogeneous earth observation satellites

ΔT_{sat} *:* satellite *sat* revolution period

P: collection of orbits/revolution/track/passes/path ρ in {1,..,$|P|$}= $|SAT|*|P_{sat}|$ (nb orbits per sat).

- Orbits are sorted in increasing order;

- $|P_{sat}| = \sup(H/\Delta T_{sat})$;
- $\rho(t, sat) = (t\text{-}1)|SAT| + sat \quad t \in \{1,2,..., H/\Delta T_{sat}\}$, $sat \in SAT$ during cycle *t;*
- sensors: electro-optical (EO), infrared (IR) or synthetic aperture radar (SAR);
- P can be partitioned in ascending and descending orbits: $P = P_A \cup P_D$; and
- *Dir:* sat direction (A/D = ascending and descending orbit).

P_{rs}: collection of orbits/revolution/track/pass ρ related to task (*r,s*). $P_{rs} \subseteq P$

T_ρ: satellite orbit ρ revolution period

R: set of candidate requests *r* (defining a target (spot) or polygon area (region) of interest (AOI) dimension A_r to be covered with beam [waveform] *B*).

- Each request *r* is decomposed into a set of tasks. A compound request includes more than one task. A spot request generally corresponds to a single primitive request involving one task. However, some primitive request/task requirement might require more than one task (e.g. a stereo task necessitating an Ascending and a Descending imaging collection (visit/observation) for suitable data analysis purposes);
- In some simple settings (e.g. single homogeneous constellation), a request (polygon area) may be decomposed into a set of sub request *s* (e.g. strips (rectangle) *s* of length l_s and width w_B defined along Earth meridian as near as polar orbit satellites are considered). Requests (*r,s,B*) are sorted increasingly by earliest starting times. Composite task examples include mosaic-based imaging, multitask mission decomposition, large area coverage (area decomposition in sub-areas);
- Task perspective:
  - R_P: set of primitive requests *r* having a single task (target) (*s*=1);
  - R_C: set of composite requests *r* having multiple tasks *s* (e.g. task package, polygons or complex structure of targets/polygons);
  - $R = R_p \cup R_c$;
  - $R_u \subseteq R_C$: set of composite requests for which individual request value contribution requires servicing all composing task *s* as a unit;
  - R_L: set of requests having piecewise linear value contributions; and
  - R_π: other complex tasks (spot target) requiring pre-defined plans.
- Resource perspective:
  - R_ρ: set of complete/partial requests (tasks) in corridor (track) visibility of orbit ρ ($R_\rho \subseteq R$) matching sensor orbit ρ capability (feasible pairing – matching). Partially visible requests are included.

S_r: set of task *s* (e.g. sub-request or strips – sub-areas/regions composing an AOI if defined accordingly) composing request *r* (or, set of subtasks *s* composing task request *r*)

[$\underline{\omega}_r$, $\overline{\omega}_r$]: completion time window for servicing (imaging) request *r* (including related tasks)

[$\underline{\omega}_{rs}$, $\overline{\omega}_{rs}$]: completion time window for servicing (imaging) request *r* and related task (assumption: $\underline{\omega}_{rs} = \underline{\omega}_r$, $\overline{\omega}_{rs} = \overline{\omega}_r$)

PREC: set of partially ordered task pairs (*r,s; r',s'*) having a precedence relationship with one another: (*r,s*) >> (*r',s'*). e.g. detection > tracking > identification; or detection>confirmation tasks, stereo imaging task or subtasks

A delay might even be necessary to account for information (observation outcome) dissemination/task flow, pre-planning and uplink (revised plan) latencies during plan execution.

ME: set of mutually exclusive task pairs

CONFL: conflicting opportunity over a task

$VIS_{rs\rho}$: binary visibility matrix indicating if request *r* task s is within sensor footprint/field of view (projected sat track area) of orbit ρ . $vis_{rs\rho}$ = 1 if (*r,s*) $\in R_\rho$: i.e. if task (*r,s*) is visible to orbit ρ (zero otherwise).

V_{r0}: nominal value of request *r*

V_{rs0}: nominal value of request-task (*r,s*). Valuation is assumed to respect utility theory axioms (large polygon areas may define a single task (not many) as many collection opportunities are required to cover the corresponding area). Apportionment of request *r* value over tasks *s.*

- $V_{r0} = \sum_s V_{rs0}$

$prop_{rs\_THR}$: Minimal coverage task (*r,s*) ratio requirement (covered area proportion – e.g. 70%)

$g_{rso\rho}$ (or $\theta_{rso\rho}$): pointing/look angle for task (*r,s*) for imaging opportunity *o* during orbit ρ

$[\underline{\theta}_{rs}, \overline{\theta}_{rs}]$: look angle intervals (near/far incidence angle) for task *s* request *r*

O: collection of all opportunities ($\cup_{rs} O_{rs}$ or $\cup_\rho O_\rho$)

O_{rs}: (collection of) opportunity (all possible orbits) for task (*r,s*). | O_{rs} | ≤ |Sat|*|P_{sat}| for non-agile sat

O_ρ: (collection of) opportunities (over all tasks) during orbit ρ. (|O_ρ |= |R_ρ | for non-agile sat)

- $O_\rho = O_\rho^A \cup O_\rho^D$
- O_ρ^A: ascending orbit direction opportunity set
- O_ρ^D: descending orbit direction opportunity set

$O_{rs\rho}$: set of collection opportunities for request *r* task *s* during orbit ρ

- *Opportunity* = {*identifier, task, B in feasible {B}*, look angle, time interval, orbit- related sensor, pass direction (Ascending/Descending), MOP [covered area, probability of success….]}
- orbit ρ implicitly determines sat direction *Dir*
- On agile satellite*:*
  - $O_{rs\rho}$={$o_{rs\rho}$(ID), *r,s*, *B*, $\boldsymbol{ts_{rso\rho}} \in [\underline{\omega}_{rso\rho}, \overline{\omega}_{rso\rho}]$, $d_{ros\rho}= te_{rso\rho} - \boldsymbol{ts_{rso\rho}} = const$, $g_{rso\rho}(ts_{rso\rho})$, *Dir*, ρ}
  - $[\underline{\omega}_{rso\rho}, \overline{\omega}_{rso\rho}]$: starting time windows for servicing subtask *(r,s)* (agile satellite) with imaging opportunity *o*
  - $g_{rso\rho}$ stands for antenna/camera pointing angle. Typically EO, not the case for SAR (rather binary left/right looking) or AIS sensor sat.
- On non-agile satellite*:*
  - $O_{rs\rho}$ = { $o_{rs\rho}$ (ID), *r,s*, *B*, $g_{rso\rho} = \theta_{rso\rho}$, [$ts_{rso\rho}$, $te_{rso\rho}$], *Dir*, ρ} $x_{rso\rho} \leq vis_{rs\rho}$
  - Reminder: $vis_{rs\rho} = 1$ if sensor orbit ρ match task (*r,s*) request requirements, otherwise $vis_{rs\rho} = 0$
  - E.g. image orientation (incidence angle) – $\theta_{rso\rho}$ in $\left[\underline{\theta}_{rs}, \overline{\theta}_{rs}\right]$, capability sensor-task, sun illumination

$TW_{rs\rho}$ = [$ts_{rs\rho}$, $te_{rs\rho}$]: time window (start time, end time) of task *s* associated with request *r* for imaging opportunity *o* during orbit ρ. For non-agile satellites, $ts_{rso\rho}$, $te_{rso\rho} < T_\rho$ are predetermined constants whereas these quantities are variables for agile satellites.

Π_{rs}: set of user-defined (sub) task plans $\pi_{rs\varphi}$ for covering (sub) task (*r,s*) ($r \in R_\pi$, $\varphi \in \Phi_{rs}$={1,2,….,|Π_{rs}|) - a (sub) task plan is a subset of collection imaging opportunities selected from a given set O_{rs} (Π_{rs} refers to a <u>family</u> of subsets of O_{rs} targeting sub task (*r,s*))

Π_{rs} = {π_{rs1}, π_{rs2}, …, $\pi_{rs\varphi}$,…, $\pi_{rs|\Pi rs|}$} = {$\pi_{rs\varphi}$} $\varphi \in \Phi_{rs}$={1,2,….,|Π_{rs}|,

$\pi_{rs\varphi}$: (sub) task plan φ defining a subset of collection imaging opportunities for covering task (*r,s*) ($r \in R_\pi$)

$\pi_{rs\varphi}$ ={$o^{\pi}_{rs\varphi 1}$, $o^{\pi}_{rs\varphi 2}$, …., $o^{\pi}_{rs\varphi|\pi\_rs\varphi|}$}

$d_{rso\rho}$: duration of task (*r,s*) imaging for opportunity *o* over orbit ρ: ($te_{rs\rho}$ - $ts_{rs\rho}$) = *const*)

Resource (time, memory, energy) capacity and consumption rates (power) per collector activity (imaging/obs, comm/downloading):

- W_ρ: memory storage capacity in orbit ρ
- w_ρ: memory consumption rate by an observation in orbit ρ
- E_ρ: energy capacity in orbit ρ

- $eo_{\rho:}$ energy consumption rate by an observation in orbit ρ
- $es_{\rho:}$ energy consumption rate for task transition by a sensor in orbit ρ
- δ_ρ *:* constant task transition time by a servicing satellite on orbit ρ (includes required time for opening (a_ρ) the sensor in orbit ρ)
- sl_ρ : slewing rate
- $\Delta t_{0\rho} + \Delta t_{rsor's'o'\rho}$ variable *task* transition time by a servicing satellite on orbit ρ from task (*r,s*) opportunity *o* to task (*r',s'*) opportunity *o'.*
  - $\Delta t_{0\rho}$: set-up time for imaging opportunity transition
  - $\Delta t_{rsor's'o'\rho}$: Transition time (e.g. $\Delta t_{rsor's'o'\rho} = | g_{rso\rho} - g_{r's'o'\rho} | / sl_\rho$)
- c_ρ : maximum number of times for a satellite opening its sensor in orbit ρ
- τ_ρ : maximal (absolute) and average acceptable imaging time over orbit ρ imposed by thermal capacity constraints
- $\bar{\tau}_{sat}$: maximal average acceptable imaging time over orbit ρ imposed by thermal capacity constraints

A_{rs}: AOI associated with task *s,* request *r*

$q_{rs\varphi}$: estimated imaging quality associated with collection plan *φ* (multiple asset opportunities) for task *s* from request *r* characterized by predetermined plans ($r \in R_\pi$, e.g. $A_{rs\varphi} / A_{rs;}$ 1-< $E_{rs\varphi}$>/$E_{rs0,}$ $rank_{rs\varphi} / rank_{max}$)

$p_{rs\varphi}$: probability to successfully execute plan *φ* (multiple asset opportunities) for task *s* from request *r* characterized by predetermined plans ($r \in R_\pi$)

$p^{succ}_{rso\rho}$: probability of success in imaging request *r* task *s* area, over opportunity *o* in orbit ρ

- e.g. an EO observing a target area under partial cloud conditions

$q_{rso\rho}$ (= $q_{rs\rho}$) : estimated imaging opportunity normalized quality associated with opportunity *o* on orbit ρ for task *s* from request *r*

- Task-dependent, account for incidence angle imaging quality, or resolution obtained for different beam mode whenever imaging entirely a spot target area, etc. e.g. an EO observing a remote area or involving a large incidence angle.

$q_{rso\rho o'\rho'}$: composed imaging opportunity quality associated with intersecting opportunity *o* and *o'* strip areas related to orbit ρ and ρ' respectively.

$q_{rso\rho o'\rho' o''\rho''}$: composed imaging opportunity quality associated with intersecting opportunity *o*, *o'* and *o"* strip areas related to orbit ρ, ρ' and ρ'' respectively.

$c_{rso\rho}$: first-order quality of information/service (utility) contribution to the collection value of task *s*, request *r*, based upon relative covered areas and observation probability of success associated with imaging opportunity *o* and orbit ρ.

- $$c_{rso\rho} = \frac{A_{rso\rho}}{A_{rs}} \underbrace{p^{succ}_{rso\rho}}_{quality\_obs \in [0,1]} q_{rso\rho}$$
  - $A_{rso\rho}$: opportunity strip area coverage of task *s,* request *r* AOI associated with opportunity *o* on orbit ρ.
  - $A^{strip}_{rso\rho}$: imaging strip area (swath) coverage associated with opportunity *o* on orbit ρ for task *s* from request *r* ($\geq A_{rso\rho}$)

$d_{rso\rho o'\rho'}$: 2nd-order quality of information/service (utility) contribution to the collection value of task *s*, request *r*, using relative mutual opportunity strip area coverage intersection and observation probability of success associated with opportunity *o* on orbit ρ, and, opportunity *o'* on orbit ρ' respectively.

- $$d_{rso\rho o'\rho'} = \frac{A_{rso\rho o'\rho'}}{A_{rs}} p^{succ}_{rso\rho} p^{succ}_{rso'\rho'} q_{rso\rho o'\rho'}$$
  - $A_{rso\rho\, o'\rho'}$: area coverage of task *s,* request *r* AOI resulting from mutual opportunity *o* and *o'* strip areas intersection related to orbit ρ, and ρ' respectively.

$e_{rso\rho o'\rho' o''\rho''}$: $d_{rso\rho o'\rho'}$: 3rd-order quality of information/service (utility) contribution to the collection value of task *s*, request *r*, using relative mutual opportunity strip area coverage intersection and observation probability of success associated with opportunity *o* on orbit ρ, opportunity *o'* on orbit ρ' , and opportunity *o"* on orbit ρ'' respectively.

- $$e_{rso\rho o'\rho' o''\rho''} = \frac{A_{rso\rho o'\rho' o''\rho''}}{A_{rs}} p^{succ}_{rso\rho} p^{succ}_{rso'\rho'} p^{succ}_{rso''\rho''} q_{rso\rho o'\rho' o''\rho''}$$
  - $A_{rso\rho\, o'\rho' o''\rho''}$: area coverage of task *s,* request *r* AOI resulting from mutual opportunity *o*, *o'* and *o"* strip areas intersection related to orbit ρ, ρ' and ρ'' respectively.

$\cos t_{rso\rho}$: *f* (swath area $A^{strip}_{rso\rho}$, *B*(*o*) , product type: resolution,…; svc provider ρ)

- e.g. $m_{rs}(B(o), resolution, \rho) A^{strip}_{rso\rho} + c_{rs0}(\rho)$

Decision variables:

- $visit_{rs}$: binary variable indicating that tasks (r,s) have been visited (at least once);
- $x_{rso\rho}$: binary variable indicating whether task s from request r is scheduled to be serviced by opportunity o on orbit ρ.
  - Many opportunities might be required;
  - Many separate tasks s compose a request r when disparate collection activities are required to fulfill r. Note in that case, that a single task s (primitive) is necessary to specify a large AOI request r, for which, multiple (possibly overlapping, or, disjoint) opportunities (and their corresponding possibly non-disjoint related strip) must be simply combined to satisfactorily cover (r,s); and
  - A plan is a set of disjoint/overlapping opportunity assigned to a task. Note that strip opportunities are simply duplicated for different opportunities emerging from identical satellite constellation members.
- $ts_{rso\rho}$: continuous variable referring to start time of tasks (r,s) for imaging opportunity o on orbit ρ
- $z_{rso\rho o'\rho'}$: binary variable indicating whether task s from request r is scheduled to be serviced by opportunity o on orbit ρ and opportunity o' on orbit ρ'
- $u_{rsor's'o'\rho}$: orbit ρ network flow binary decision variable indicating a transition between task s under request r opportunity o and task s' under request r' opportunity o' in orbit ρ (i.e. (r',s') will be executed after (r,s))
  - number u var $\leq \sum_{\rho} \frac{1}{2} (\sum_{r,s} ||O_{rs\rho}||)^2$
- Piecewise linear objective function contribution over request r :
  - The objective function contribution for request $r \in R_L$ is linearly defined over v_r ($m_{ir} v_r + b_{ir}$). Ranging in [0,1] v_r is divided in pre-defined intervals i from a set I_r
  - q_{ir} : binary variable indicating interval i of the piecewise linear objective function value contribution for servicing request r
  - l_{ir}: continuous variable over [0,1] reflecting piecewise linear contribution on interval i to the value of request r
- All task service requirements under request $r \in R_u \subseteq R$
  - wu_r: binary variable reflecting all task s service requirements for task r impacting the value of request r
  - X_{rs}: binary variable indicating if task s from request r is serviced
  - v_r: continuous variable over [0,1] capturing the value of request r
  - $w_{rs\varphi}$: binary plan variable referring to the selection of plan φ among possible plans in the set Π_{rs} to cover task (r,s). The null plan is included. $\varphi \in \Phi_{rs} \cup \{0\}$

3.6.2 Constraints and Objective

These following constraints are taken into account when building the Graph network.

Transition: (agile sat (time windows); non-agile sat have fixed *ts* values [unnecessary]).

$$ts_{r's'o'\rho} + M(1 - u_{rsor's'o'\rho}) \geq \underbrace{ts_{rso\rho} + d_{rso\rho}}_{te_{rso\rho}} + \left(\overbrace{\Delta t_{0\rho} + \Delta t_{rsor's'o'\rho}}^{transition\_time} \right) \quad M >> 1$$

- Note: if time windows are discretized in the acyclic graph, all opportunities have a fixed time and that constraint is unnecessary.

Time window constraints:

$\underline{\omega}_{rso\rho} x_{rso\rho} \leq ts_{rso\rho} \leq \overline{\omega}_{rso\rho} x_{rso\rho}$ Opportunity time window

$\underline{\omega}_{rs} x_{rso\rho} \leq ts_{rso\rho} + d_{rso\rho} x_{rso\rho} \leq \overline{\omega}_{rs} x_{rso\rho}$ Completion time window for request *r* and related task *s*

$$\left(\left(\frac{\rho - \rho \bmod |SAT|}{|SAT|} \right) T_\rho \leq \underline{\omega}_{rso\rho}, \overline{\omega}_{rso\rho} < \left(\frac{\rho - \rho \bmod |SAT|}{|SAT|} + 1 \right) T_\rho \right)$$

$$\left(\underline{\omega}_{rso\rho} = \overline{\omega}_{rso\rho} < \left(\frac{\rho - \rho \bmod |SAT|}{|SAT|} + 1 \right) T_\rho ; \quad ts_{rso\rho} + d_{rso\rho} = te_{rso\rho} \right)$$ For non-agile satellite

Objective function:

$$MAX \quad \sum_{r \in R} \sum_{s \in S_r} \left(V_{r0} \frac{A_{rso\rho}}{A_r} q_{rso\rho} \left(1 - \prod_{\rho \in P_{rs}} \left(1 - p_{rso\rho} x_{rso\rho} \right) \right) \right) \tag{1}$$

$$\sum_{r'}^{|R|} \sum_{s'}^{|S_r|} \sum_{o'} u_{rsor's'o'\rho} = x_{rso\rho} \tag{2}$$

Flow conservation:

$$\sum_{r}^{|R|} \sum_{s}^{|S_r|} \sum_{\substack{o \\ ((rso),(r's'o')) \in A_\rho \\ G_\rho(V_\rho, A_\rho)}} u_{rsor's'o'\rho} - \sum_{r''}^{|R|} \sum_{s''}^{|S_r|} \sum_{\substack{o'' \\ ((r's'o'),(r''s''o'')) \in A_\rho \\ G_\rho(V_\rho, A_\rho)}} u_{r's'o'r''s''o''\rho} = 0 \quad \forall r' \in R, s' \in S_r, o' \in O_{r's'\rho}, \rho \in P \tag{3}$$

Energy Constraint:

$$\sum_{r\in R}\sum_{s\in S_r}\sum_{o\in O_{rs\rho}} x_{rso\rho}\, eo_\rho\,(te_{rso\rho}-ts_{rso\rho})+$$

$$\sum_{r\in R\cup\{b\}}\ \sum_{\substack{s\in S_r\\ ((rso),(r's'o'))\in A_\rho\\ G_\rho(V_\rho,A_\rho)}}\ \sum_{o\in O_{rs\rho}}\ \sum_{r'\in R\cup\{e\}}\ \sum_{s'\in S_{r'}}\ \sum_{o'\in O_{r's'\rho}} u_{rso r's'o'\rho}\, es_\rho\left(\Delta t_{0\rho}+\Delta t_{rso r's'o'\rho}\right)\le E_\rho \quad \forall\rho\in P \tag{4}$$

Memory storage capacity:

$$\sum_{r}^{|R|}\sum_{s}^{|S_r|}\sum_{o\in O_{rs\rho}} w_\rho \underbrace{\left(te_{rso\rho}-ts_{rso\rho}\right)}_{d_{rso\rho}} x_{rso\rho}\le W_\rho \quad \forall\rho\in \mathrm{P} \tag{5}$$

Max number of sensor openings c_ρ by orbit: orbit capacity

$$\sum_{r}^{|R|}\sum_{s}^{|S_r|}\sum_{o\in O_{rs\rho}} x_{rso\rho}\le c_\rho \quad \forall\rho\in \mathrm{P} \tag{6}$$

Image acquisition costs: financial capacity (cost is proportional to resolution x coverage)

$$\sum_{r}^{|R|}\sum_{s}^{|S_r|}\sum_{\rho}^{|\mathrm{P}|}\sum_{o\in O_{rs\rho}} \cos t_{rso\rho}\; x_{rso\rho}\le \cos t_{\max} \tag{7}$$

Maximum Image acquisition cost: financial capacity per task

$$\sum_{\rho}^{|\mathrm{P}|}\sum_{o\in O_{rs\rho}} \underbrace{\cos t_{rso\rho}}_{m_{rs(B,\mathrm{inf}o\_product)}\, A^{strip}_{rso\rho}+c_{rs0}} x_{rso\rho}\le \cos t^{\max}_{rs} \tag{8}$$

Thermal constraints (max imaging time by orbit; average max imaging time by orbit):

$\rho(t, sat) = (t\text{-}1)\,|SAT| + sat \quad t\in\{1,2,\ldots, H/\Delta T_{sat}\},\ sat\in SAT$

$$\sum_{r}^{|R|}\sum_{s}^{|S_r|}\sum_{o\in O_{rs\rho}} d_{rso\rho}\, x_{rso\rho}\le \tau_\rho \quad \forall\rho\in \mathrm{P} \tag{9}$$

$$u_{rso r's'o'\rho}, x_{rso\rho}\in\{0,1\} \tag{10}$$

The objective (1) is to maximize the expectation value of the profits of the executed tasks under uncertainties of clouds, in which $\left(1-\prod_{\rho\in P_{rs}}\left(1-p_{rso\rho}x_{rso\rho}\right)\right)$ denotes the completion probability for task (r,s) when it is scheduled to multiple orbits and this quantity is multiplied by a weight which combines the quality, priority and the proportion of the area covered by the opportunity. The set of constraints (2) indicate if the task (r,s) is visited or not (or covered by an opportunity). The set of constraints (3) that are flow conservation constraints ensure that the number of predecessors is equal to the number of successors for each task. Constraints (4) compute the energy consumption of the task sequence for each orbit, and enforce that the energy consumption must be less than or equal to the capacity. Constraints (5) check that the memory consumption of the scheduled tasks cannot exceed the memory capacity for each orbit. Constraints (6) check that the task sequence for each orbit does not exceed the given capacity of the orbit. Constraints (7) specify that the total tasks covered by all orbits do not exceed the total budget available. Constraints (8) specify that the sequence of tasks covered by an orbit does not exceed the total budget available for this task. Constraints (9) compute the time consumption for imaging task sequence for each orbit is less than or equal to the available time dedicated to image (image capacity). Finally, constraints (10) set that all implicated variables are integers.

3.7 QUEST

Unfortunately, the master problem above (objective function 1) is still neither linear nor quadratic, and thus we cannot apply successfully existing solvers or algorithms.

Taking advantage of the problem structure, the first remark is that for a given task, a limited number of visits are enough to make the probability of success close to one. This leads to an approximate model in which we limit the number of visits to 1, 2 or 3 depending on the initial probability of success. Also, we divided tasks into two sets of tasks: tasks with plans, we modeled it as coarse-grained model and tasks without plans; we modeled it as a fine-grained model. In the first model we keep the constraints above (2-10) and only the objective function is changed to as follows:

$$MAX \sum_{r\in R=R_C\cup R_P} v_r = \sum_{r\in R_C} v_r + \sum_{r\in R_P} v_r$$

Subject to:

Complex task (coarse-grained modeling):

$$v_r \le \sum_{s\in S_r}\sum_{\varphi\in\Phi_{rs}} V_{rs0}q_{rs\varphi}p_{rs\varphi}w_{rs\varphi} = \sum_{s\in S_r} V_{rs0}\overbrace{\sum_{\varphi\in\Phi_{rs}} \underbrace{q_{rs\varphi}}_{\substack{precomputed\\ for\ each\ \pi_{rs\varphi}}} p_{rs\varphi}w_{rs\varphi}}^{QoI_{rs}=utility_{rs}} \qquad r\in R_\pi$$

$$w_{rs\varphi} \le \frac{1}{\left\|\pi_{rs\varphi}\right\|}\sum_{\rho\in P_{rs}}\sum_{o\in O_{rs\rho}\cap\pi_{rs\varphi}} x_{rso\rho} \quad r\in R_\pi, s\in S_r, \varphi\in\Phi_{rs}=\{1,2,...,\left|\Pi_{rs}\right|\}, \pi_{rs\varphi}\in\Pi_{rs}$$

$$\sum_{\varphi\in\Phi_{rs}} w_{rs\varphi} \le 1 \quad \forall r\in R_\pi, s\in S_r$$

Complex task (fine-grained modeling):

$$v_r \le \sum_{s\in S_r} V_{rs0}\{\sum_{\rho\in \mathrm{P}_{rs}}\sum_{o\in O_{rs\rho}} c_{rso\rho}x_{rso\rho} - \sum_{\rho\in \mathrm{P}_{rs}}\sum_{\substack{o\in O_{rs\rho}\\(\rho,o)<(\rho',o')}}\sum_{\rho'\in \mathrm{P}'_{rs}}\sum_{o'\in O_{rs\rho'}} d_{rso\rho o'\rho'}\left[\frac{\left(x_{rso\rho}+x_{rso'\rho'}\right)^2-\left(x_{rso\rho}+x_{rso'\rho'}\right)}{2}\right]+$$

$$\sum_{\rho\in \mathrm{P}_{rs}}\sum_{o\in O_{rs\rho}}\sum_{\substack{\rho'\in \mathrm{P}'_{rs}\\(\rho,o)<(\rho',o')<(\rho'',o'')\to(\rho,o),(\rho',o'),(\rho'',o'')\in SET}}\sum_{o'\in O_{rs\rho'}}\sum_{\rho''\in \mathrm{P}''_{rs}}\sum_{o''\in O_{rs\rho''}} e_{rso\rho o'\rho' o''\rho''}\left[\frac{\left(z_{rso\rho o'\rho'}+x_{rso''\rho''}\right)-\left(z_{rso\rho o'\rho'}-x_{rso''\rho''}\right)^2}{2}\right]-....\}$$

$$r\in R\,|\,R_\pi$$

$$z_{rso\rho o'\rho'} \le x_{rso\rho} \qquad r\in R\,|\,R_\pi$$

$$z_{rso\rho o'\rho'} \le x_{rso'\rho'} \qquad r\in R\,|\,R_\pi$$

$$z_{rso\rho o'\rho'} \ge x_{rso\rho}+x_{rso'\rho'}-1 \qquad r\in R\,|\,R_\pi$$

$$c_{rso\rho}=\frac{A_{rso\rho}}{A_{rs}}\underbrace{p^{succ}_{rso\rho}}_{quality\_obs\in[0,1]}\underbrace{\overbrace{q_{rso\rho}}^{qual\in[0,1]}}_{Q_{1rso\rho}},\quad d_{rso\rho o'\rho'}=\frac{A_{rso\rho o'\rho'}}{A_{rs}}p^{succ}_{rso\rho}p^{succ}_{rso'\rho'}Q_{2rso\rho o'\rho'},\quad e_{rso\rho o'\rho' o''\rho''}=\frac{A_{rso\rho o'\rho' o''\rho''}}{A_{rs}}p^{succ}_{rso\rho}p^{succ}_{rso'\rho'}p^{succ}_{rso''\rho''}Q_{3rso\rho o'\rho' o''\rho'''}$$

$$Q_{2rso\rho o'\rho'}=\left(q_{rso\rho}+q_{rso'\rho'}\right)-q_{rso\rho o'\rho'},\qquad Q_{3rso\rho o'\rho' o''\rho'''}=\left(q_{rso\rho}+q_{rso'\rho'}+q_{rso''\rho''}\right)-\left(q_{rso\rho o'\rho'}+q_{rso\rho o''\rho''}+q_{rso'\rho' o''\rho''}\right)+q_{rso\rho o'\rho' o''\rho''}$$

$$note: q_{\bullet}=1\Rightarrow Q_{1,2,3,...}=1.\quad \textit{In addition, if } p^{succ}_{\bullet}=1 \textit{ a unique visit is required for a spot TARGET}!!!$$

$$(A^{cvg*}_{rs}\approx \max_{\{x_{rso\rho}\}}\sum_{\rho}\sum_{o\in O_{rs\rho}}\frac{A_{rso\rho}}{A_{rs}}x_{rso\rho}-\sum_{\rho}\sum_{\substack{o\in O_{rs\rho}\\(\rho,o)<(\rho',o')''}}\sum_{\rho'}\sum_{o'\in O_{rs\rho'}}\frac{A_{rso\rho o'\rho'}}{A_{rs}}\left[\frac{\left(x_{rso\rho}+x_{rso'\rho'}\right)^2-\left(x_{rso\rho}+x_{rso'\rho'}\right)}{2}\right]+$$

$$\sum_{\rho}\sum_{o\in O_{rs\rho}}\sum_{\rho'}\sum_{\substack{o'\in O_{rs\rho'}\\(\rho,o)<(\rho',o')<(\rho'',o'')\to(\rho,o),(\rho',o'),(\rho'',o'')\in SET}}\sum_{\rho''}\sum_{o''\in O_{rs\rho''}}\frac{A_{rso\rho o'\rho' o''\rho''}}{A_{rs}}\left[\frac{\left(z_{rso\rho o'\rho'}+x_{rso''\rho''}\right)-\left(z_{rso\rho o'\rho'}-x_{rso''\rho''}\right)^2}{2}\right]-....)$$

3.8 RE-CVEST

QUEST Model quickly reaches its capacity. In the case of coarse-grained model the capacity is quickly reached because of the exponential number of possible plans. As each variable corresponds to a plan this leads to a model of exponential variables thus very difficult to solve. As for the fine-grained model, it leads to a limited number of variables and the fact that multiple visits to a task lead to both positive and negative contributions in the objective function. This fact has the effect of causing the relaxed solution to be far from the MIP solution. This is the reason why we have developed a new model that exploits the advantages of both previous models while avoiding inconvenient of both approaches. This allowed us to succeed in solving bigger problems.

We develop a new mathematical model which provides good upper bound and can solve large size combinatorial optimization problems. This model is derived from the quadratic model by introducing new variables which find how many times a given task is visited (this maximum number of visits is limited to 2-3) and add the right positive contribution to the objective function. This model is based on delayed-reward approach which does not impose any number of visits on a given task but reward only the first 1 or 2 or 3 visits. The remained visits are rewarded correctly only on the final solution. The constraints 2 to 10 are unchanged; the objective function is transformed as follows respectively on the case of two or three visits:

$$\max \quad \sum_r V_{r0} \left[\begin{array}{l} \sum_{i \in O} \frac{A_{ri}}{A} p_{ri} y_{ri} + \sum_{i<j \in O} \left(p_{ri} \frac{A_{ri}}{A} + p_{rj} \frac{A_j}{A} - p_{ri} p_{rj} \frac{A_{rij}}{A} \right) z_{rij} + \\ \sum_{i<j<k \in O} \left(p_{ri} \frac{A_{ri}}{A} + p_{rj} \frac{A_{rj}}{A} + p_{rk} \frac{A_{rk}}{A} - p_{ri} p_{rj} \frac{A_{rij}}{A} - p_{ri} p_{rk} \frac{A_{rik}}{A} - p_j p_k \frac{A_{jk}}{A} + p_{ri} p_{rj} p_{rk} \frac{A_{rijk}}{A} \right) w_{rijk} \end{array} \right]$$

s.t.

$$y_i \le x_i \quad \forall i \in O_{rs} \quad \text{sin}\, gle\_visit$$

$$\sum_{i \in O_{rs}} y_i + \sum_{i<j \in O_{rs}} z_{ij} + \sum_{i<j<k \in O_{rs}} w_{ijk} \le 1 \quad \forall r,s \in R \quad \left(\sum_{i \in O_{rs}} y_i + \sum_{i<j \in O_{rs}} z_{ij} \le 1 \quad \forall i \in O_{rs} \quad for\ two\ \ visits \right) \quad for\ \ 3\ visits$$

$$z_{ij} \le \frac{1}{2}\left(x_i + x_j\right) \quad \forall i<j \in O_{rs}, \quad w_{ijk} \le \frac{1}{3}\left(x_i + x_j + x_k\right) \quad \forall i<j<k \in O_{rs}$$

3.9 Methodology and Solution Approach

It is well known that linear programming problems and integer programming problems are very difficult to solve. In fact, none efficient general algorithm is known for their solution. Given our inability to solve integer programming problems efficiently, it is natural to ask whether such problems are inherently “hard”. Complexity theory overcomes insight on this question. It provides us with a class of problems with the following property: if a polynomial time algorithm exists for any problem in this class, then all integer programming problems can be solved by a polynomial algorithm, but this is considered unlikely. Algorithms for integer programming problems rely on two basic concepts: Relaxation and Branch-and-Bound. There are three main categories of algorithms for integer programming problems:

- Exact algorithms that guarantee to find an optimal solution, but may take an exponential number of iterations. They include cutting planes, branch-and-bound, and dynamic programming;
- Heuristic algorithms that provide a suboptimal solution, but without a guarantee on its quality. Although the running time is not guaranteed to be polynomial, empirical evidence suggests that some of these algorithms find a good solution faster; and
- Approximation algorithms that provide in polynomial time a suboptimal solution together with a bound on the degree of sub-optimality.

The reason why a Mixed Integer Linear Problem is hard to solve lies in the fact that the gap is very large (see Figure 2). That is the case when there are many competitions between variables in the objective value that lead to a Lagrangian relaxation which is a far from feasible integer solution. Due to the exponential growth in the size of such a tree, exhaustive enumeration would quickly become hopelessly computationally expensive for MIPs.

By examining the output of the branch-and-bound algorithm, one can often identify the cause(s) of the performance problem.

For our problem QUEST, we made the following remarks:

i) The first Lagrangian relaxation linear problem solution U0 is very bad and very far from the MIP solution;

ii) Fortunately, the second phase iteration in the RE-CVEST model, CPLEX provides very tight upper-bound U2 by using Zero-half cuts. *Zero-half cuts* are based on the observation that when the left-hand side of an inequality consists of integral variables and integral coefficients, then the right-hand side can be rounded down to produce a zero-half-cut. By using intensively these cuts we get a very tight upper bound;

iii) Each time CPLEX adds a cut, the sub-problem is re-optimized. ILOG CPLEX repeats the process of adding cuts at a node until it finds no further effective cuts;

iv) This upper-bound is used in the feasible solution search phase as a heuristic. This heuristic consists of setting to zero any variable which is close to zero in the relaxation in U2. This setting is done via the callable library of CPLEX by using our own heuristic algorithm. And solve again the derived sub-problem. This leads to a very good solution value where the gap is close to zero. Finally, we leave MIP branch-and-cut algorithm achieves the remaining steps of MIP solving; and

v) Depending on how hard is the initial problem (kSAT is close to 10), we impose a stopping criterion (gap <=5%).

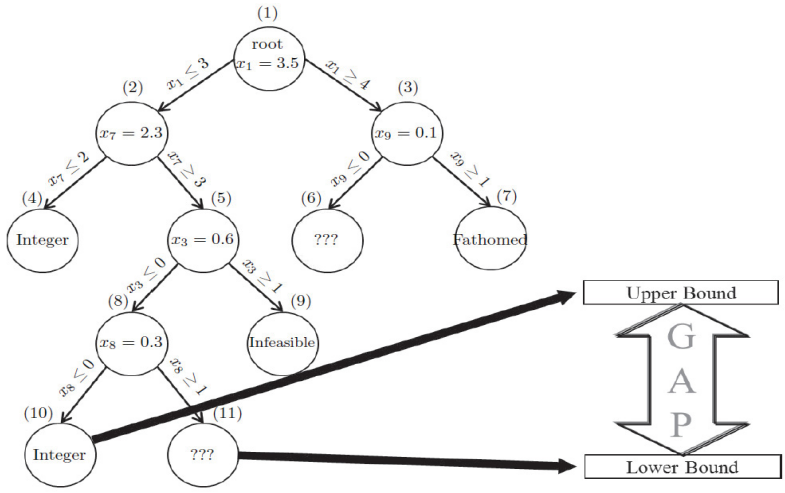


Figure 2: Branch-and-bound Algorithm

3.10 Large Area Coverage Problem – MOSAIC

This mathematical model consists of a particular case of QUEST/RE-QUEST model where the number of tasks is equal to one and that task is very large. With the difficulty to solve this problem in many situations, we designed a solution methodology which solves this problem with a gap less than or equal to 5%. The approach consists of 2 steps which are called set covering and blocks holes method (SCBH):

1. Solve a first linear model which consists of covering the largest area with non-overlapping spot targets as much as possible. If *o* and *o'* overlap, then add:

$$x_{rso\rho} + x_{rso'\rho'} \leq 1 \text{ (overlap free constraint) } \forall r \in R, s \in S_r, \rho, \rho' \in P, o \in O_{rs\rho}, o' \in O_{rs\rho'}$$

With this objective function:

$$\max_{\{x_{rso\rho}\}} \sum_{\rho} \sum_{o \in O_{rs\rho}} \frac{A_{rso\rho}}{A_{rs}} x_{rso\rho}$$

2. Remove all overlap free constraints. And solve a second linear model which blocks the remaining holes while minimizing overlapping.

$x_{rs\rho} = 1$, obtained from the first step.

$$(A_{rs}^{cvg*} \approx \max_{\{x_{rso\rho}\}} \sum_{\rho} \sum_{o \in O_{rs\rho}} \frac{A_{rso\rho}}{A_{rs}} x_{rso\rho} - \sum_{\rho} \sum_{\substack{o \in O_{rs\rho} \\ (\rho,o)<(\rho',o')"}} \sum_{\rho'} \sum_{o' \in O_{rs\rho'}} \frac{A_{rso\rho o'\rho'}}{A_{rs}} \left[\frac{\left(x_{rso\rho} + x_{rso'\rho'}\right)^2 - \left(x_{rso\rho} + x_{rso'\rho'}\right)}{2} \right] +$$

$$\sum_{\rho} \sum_{o \in O_{rs\rho}} \sum_{\rho'} \sum_{\substack{o' \in O_{rs\rho'} \\ (\rho,o)<(\rho',o')<(\rho",o") \rightarrow (\rho,o),(\rho',o'),(\rho",o") \in SET}} \sum_{\rho"} \sum_{o" \in O_{rs\rho"}} \frac{A_{rso\rho o'\rho' o"\rho"}}{A_{rs}} \left[\frac{\left(z_{rso\rho o'\rho'} + x_{rso"\rho"}\right) - \left(z_{rso\rho o'\rho'} - x_{rso"\rho"}\right)^2}{2} \right] - ....)$$

We use the natural upper bound of this problem (by determining the effective area covered by spot targets). We show that we can solve very large size problems (more than 1000 spot targets) with a gap less than or equal to 5%.

3.11 Results, Tests and Comparisons

The features or aspects of the mathematical model that is novel, useful and not obvious:

- Some constraints are surplus, i.e. we can remove many of them and keep only one with a tighter upper bound which reduces the complexity and size of the problem definition;
- The objective function is not linear but depending on the probability of taking the image under uncertainty (cloud) a maximum number of three visits are sufficient to reduce the uncertainty and increase the probability of success close to one. In many cases this maximum number of visits can be limited to two or one. This remark reduces considerably the size of the combinatorial space and makes the problem solvable;
- A network acyclic graph is used in order to reduce the number of variables and to speedup branch-and-bound algorithmic phase;
- Highly flexible through user definition of complex tasks and resource usage: recurring and non-recurring tasks, resource constraints, set-up times, predecessor constraints, etc.; and
- To address large problem instances, different strategies are used, for example by fixing or removing non-promising opportunities (which are not interesting anymore in the Lagrangian relaxation, i.e. integrality constraints and the very difficult energy constraints are relaxed) in order to prune and reduce the large feasible space of very large size problem instances.

Wang data validation with small size instances (our results are compared to Wang exact solution): In all problem instances, QUEST found a solution in a few seconds (less than 30 seconds) with a gap less than 1%. To evaluate our model and algorithm, we performed experiments on Wang data and by using our own data generator and compared the scheduling schemes generated by different methods namely MY-PICC and GATHER fast and slow. The results of experimental simulations validated the impact (in terms of the complexity) of the total number of times a task is visible. The comparisons and analysis performed in this study demonstrated that the total number of times a task is visible by satellites is an important and main factor which determines the solvability. When the total number of times a task is visible by satellite is less than or equal to 10, our approach which is based on a barrier algorithm can improve the robustness of the produced schedules.

3.12 Limitations of the Approach

Because this problem is a very hard constrained resource combinatorial optimization problem, our experimental simulation with RE-CVEST shows that our solution approach has a natural limitation when we considered more than 300 tasks with more than 10 opportunities for each of them. This means that we have plenty of tasks which are visible to many of satellites.

Computational study 1:

- 40 problem instances provided by Wang data input requests;
- Stopping criteria: optimality (gap = 0); and
- Solutions for most of the instances within 1% of the upper bound in less than 1 minute.

Computational study 2:

- 100 problems instances provided by Berger data input requests (randomly generated);
- Stopping criteria: optimality (gap <= 5%); and
- Solutions for most of the instances within 5% of the upper bound in less than 5 minutes.

Table 2: Performance comparisons of QUEST for 6 or 12 orbits and 250, 500 and 1000 tasks and kSAT factor less than or equal to 6

Generated Seed	m = #Orbits	n = #Tasks	O = # Opportunities	Total arcs	Building time	alpha	Exact FG Quad 2 visits	Solving Time CG 2 visits	k*SAT factor	ϱ effective factor	Gap	Objective CG
1493006714583	12	500	1500	59836	12	0.150	688.7130561	57	3	12	0.00%	688.713056
1493004630803	12	250	1500	59465	1	0.015	544.6127883	214	6	12	0.00%	544.612788
1492953320943	6	500	1500	NR*	10	0.020	1156.020907	39	6	6	0.00%	1156.02091
1492933203011	6	500	1500	NR	12	0.025	989.1996841	37	6	6	0.00%	989.199684
1492932237065	6	500	1500	NR	15	0.030	904.6396283	32	6	6	0.00%	904.639628
1492930076322	6	500	1500	NR	11	0.050	634.3546341	35	6	6	0.00%	634.354634
1492929033313	6	500	1500	NR	13	0.250	162.3413539	16	6	6	0.00%	162.341354
1492929542279	6	500	1500	NR	13	0.450	85.15018363	3	6	6	0.00%	85.1501836
1492929724282	6	500	1500	NR	19	0.500	57.97508612	3	6	6	0.00%	57.9750861
1493223808933	12	500	3000	294823	17	0.1	743.9130024	257	6	12	0.00%	743.913002
1492719687575	6	500	3000	548232	13	0.010	45.30914432	530	6	12	0.00%	45.3091443
1492719687575	6	250	1500	113045	2	0.025	27.95717244	71	6	6	0.00%	27.9571724
1492719687575	6	250	1500	113045	2	0.150	27.95717244	76	6	6	0.00%	27.9571724
1493523080460	12	250	1500	56004	5	0.150	43.44568539	41	6	12	0.00%	43.4456854
1493525357851	12	500	6000	225236	13	0.150	54.77417403	204	6	12	0.00%	54.774174
1493529401004	12	1000	12000	895758	45	0.150	67.90010051	933	6	12	0.00%	67.9001005
1493545451901	12	1000	12000	NR	47	0.100	104.9023064	973	6	12	0.00%	104.902306
1493556869806	12	500	6000	274440	17	0.450	75.1605459	624	6	12	0.00%	75.1605459

*NR: not reported.

Table 3: Performance comparisons of QUEST for 15 orbits, 300 tasks and kSAT factor equal to 8

#GeneratedSeed	#number of opportunities	#kSAT*	# orbits Effectives	# tasks	#number of arcs or	#run time QUEST	#Objective MY-PICC	#Objective Gather	#Run time gather long	#Gather Long	#objective QUEST	#Upper Bound	#GAP MY PICC	#GAP Gather	#GAP Gather long time	#GAP QUEST	#Images Number Gather Long	#Image Number QUEST	QUEST/Gather Long / images	#Increase revenu/day	#alpha
15018794493551	2400	8	15	300	184782	107	109.447839	111.944724	288	135.843142	159.7282	163.7228	33.15%	31.63%	17.03%	2.44%	387.00	443.00	56.00	$28,000.00	0.01
15018815344211	2400	8	15	300	184820	73	119.902508	112.663184	270	139.216742	159.249093	165.2956	27.46%	31.84%	15.78%	3.66%	366.00	428.00	62.00	$31,000.00	0.01
15018818235871	2400	8	15	300	185039	48	112.194945	111.386113	113	137.852577	161.165166	167.78	33.13%	33.61%	17.84%	3.94%	348.00	413.00	65.00	$32,500.00	0.01
15018819756441	2400	8	15	300	184911	39	119.451811	110.90805	210	132.934511	154.244809	159.5963	25.15%	30.51%	16.71%	3.35%	339.00	422.00	83.00	$41,500.00	0.01
15018820835671	2400	8	15	300	185071	59	119.244124	114.375347	343	140.328832	161.503721	167.4306	28.78%	31.69%	16.19%	3.54%	374.00	430.00	56.00	$28,000.00	0.01
15018823363381	2400	8	15	300	184784	45	113.153547	109.053871	204	138.529061	160.512438	165.8039	31.75%	34.23%	16.45%	3.19%	365.00	423.00	58.00	$29,000.00	0.01
15018824482641	2400	8	15	300	184923	53	101.998226	104.076205	256	133.522804	153.266718	159.4298	36.02%	34.72%	16.25%	3.87%	364.00	436.00	72.00	$36,000.00	0.01
15018828655811	2400	8	15	300	184705	55	124.276881	109.244315	267	141.849538	163.481169	169.7147	26.77%	35.63%	16.42%	3.67%	365.00	424.00	59.00	$29,500.00	0.01
15018829930161	2400	8	15	300	184521	63	111.718723	113.425449	246	132.229069	153.356739	159.1974	29.82%	28.75%	16.94%	3.67%	364.00	420.00	56.00	$28,000.00	0.01
15018831992481	2400	8	15	300	185185	73	104.477192	104.644075	119	127.997223	152.236513	156.883	33.40%	33.30%	18.41%	2.96%	343.00	404.00	61.00	$30,500.00	0.01

Table 4: Performance comparisons of QUEST for 15 orbits, 500 tasks and kSAT factor equal to 8

#GeneratedSeed	#number of opportunities	#kSAT*	# orbits Effectives	# tasks	#number of arcs or	#run time QUEST	#Objective MY-PICC	#Objective Gather	#Run time gather long	#Gather Long	#objective QUEST	#Upper Bound	#GAP MY PICC	#GAP Gather	#GAP Gather long time	#GAP QUEST	#Images Number Gather Long	#Image Number QUEST	QUEST/Gather Long / images	#Increase revenu/day	#alpha
15018794493551	4000	8	15	500	509109	257	164.838488	138.398551	174	175.180947	237.331536	248.0979	33.56%	44.22%	29.39%	4.34%	409.00	510.00	101.00	$50,500.00	0.01

Table 5: Performance comparisons of RE-CVEST for 15 orbits, 750 tasks and kSAT factor equal to 10 (Harder instances)

#GeneratedSeed	#number of opportunities	#kSAT*	# orbits Effectives	# tasks	#number of arcs	#run time	#objective	#GAP
15014389180801	7500	10	15	750	1830441	344	3958.5331	0.0137
15014702038931	7500	10	15	750	1830904	173	3853.1171	0.0227
15014713927491	7500	10	15	750	1831088	204	3970.36076	0.0122
15014716734791	7500	10	15	750	1831776	319	3595.79396	0.0493
15014720767721	7500	10	15	750	1831734	467	3885.10365	0.0385
15014726642281	7500	10	15	750	1831102	247	3845.90106	0.0182
15014729896821	7500	10	15	750	1830492	429	3835.21393	0.0149
15014736104191	7500	10	15	750	1830811	414	3866.19853	0.0341
15014741434331	7500	10	15	750	1832470	416	3819.26637	0.0149
15014748936901	7500	10	15	750	1832298	517	3884.60594	0.0216

3.13 Conclusion

In this section, by modeling mathematically the uncertainties of clouds, we formulated the presence of clouds as stochastic events. Due to these possible uncertainties of clouds, a task that is scheduled (may be more than one) will be completed at the highest probability of success as possible (close to one). This fact leads to an original and first model which is not linear. We exploited the structure of the problem in order to make it linear in our first approximated model Coarse- and Fine-grained (CFG). This CFG model is very good but has big limitations and fails in solving large size instances. The reason is that Coarse-grained model needs plenty of integer variables (due to the presence of a large number of possible plans) while Fine-Grained model must consider overlaps between area. The penalty of this overlaps creates a competition between the integer variables. This has the effect of artificially increasing the gap and makes the problem very hard to solve. In the last mathematical model (RE-CVEST model), we eliminated this competition and we managed to solve problems by exploiting the information contained in the Lagrangian relaxation problem (which we know that its solution is a good upper bound because there is no competition between integer variables). We use a heuristic in order to speed-up solution finding by setting to zero all variables which are close to zero in the Lagrangian relaxation problem. We also use Barrier algorithm which is proven to be the best algorithm to solve Lagrangian relaxation problems. Also, we use Branch-and-cut (-and-price) algorithms which belong to the most successful techniques for solving mixed integer linear programs and combinatorial optimization problems to optimality (or, at least, with certified quality).

If a task is visited more than 3 times, the contribution of next visits is very small to the point that it can be ignored in the original model. The last mathematical model (RE-CVEST model) is the best model which can solve very large size problem and combinatorial problems (more than 500 tasks) in less than 5 minutes with a gap less than or equal to 5%. The Lagrangian relaxation also provides a very good upper-bound of this problem. From the simulation experiments, we verified the superiority of our robust last model compared with the heuristics which provides a solution with no idea of the solution quality and the upper-bound. Finally, we compared RE-CVEST with five heuristics. The solution of RE-CVEST ranked better by more than 15% and in some instances 30% (compared to the best of these 5 heuristics).

In the first step of our MOSAIC approach, which consists of two steps Mixed Integer Linear Model, we added a batch of constraints which allow us to cover the largest area with no overlap areas (i.e. overlap between selected opportunities equals zero for every two sub-area in the solution). Finally, we tackled the problem in the second phase by filling all the remaining holes created by fixing some opportunities we find in the first step (solution) with the minimum overlap with these fixed opportunities. The experimental result shows that we can solve a very large area with a very tight gap fewer than 5% compared to the upper-bound (this upper-bound is given by the largest area we can cover by using all opportunities).

3.14 Future Work

Following from this work, the following steps are required to develop a fully functional and operational *DISCOVER*:

- Explore new algorithms in order to solve very large size and harder instances (up to 10,000 opportunities and up to 1000 tasks with up to 10 multi-visits on many tasks); and
- Providing start lp solution (from GATHER for example) of CPLEX in order to reduce runtime.

Future work could account for virtual constellation of non-trailing heterogeneous satellites and other types of platforms (e.g. unmanned aerial vehicles), additional mission/operation constraints (e.g. mandatory task, task precedence) and multiple objectives in order to enrich the problem and make it more realistic. An alternate research direction aims at exploring dynamic re-tasking, building upon the anytime property naturally shown by genetic algorithms. Finally, we have to revisit the model in order to take account downlink scheduling constraints.

4 HEURISTICS MODELS/SOLVERS

4.1 MY-PICC - Time-weighted variant

4.1.1 Description

MY-PICC is a *MYopic Planning-based Image aCquisition heuristiC*.

This simple heuristic is inspired from Verfaillie et al. [10]. It is a greedy heuristic to select collection opportunities (matches between imaging tasks and platforms/sensors that can perform it) from per-orbit opportunity graphs.

In accordance with our data model, satellites are hereafter referred to as platforms.

Here are heuristic key points and definitions to complement the pseudo-code in section 4.1.2.

4.1.1.1 Time Frame

- All times are relative to the time horizon, i.e. within **[0.0, horizon[**:
  - Until this heuristic is applied on real data, this is a fictitious time frame; and
  - With real data, time 0.0 could be “now” for instance, or any other user-specified reference.
- This heuristic divides the time frame into orbits:
  - For now, all platforms are assumed to have the same revolution period.

4.1.1.2 Opportunity Graph

- Vertices are the set of opportunities for a specific platform that *start* within a given time frame (here an orbit duration);
- Edges are directed and are the set possible transitions from an opportunity to another:
  - When **building** the graph between opportunities A and B, only the time constraint determine feasibility:
    - **B start time >= A end time + AB transition duration + B set-up time.**
  - When **running the heuristic,** other constraints (energy, memory, cost, etc.) can make an edge infeasible. Constraints are essentially found in the data model, see section 1.1.7 Constraints set. Calculation of constrained value may also depend on already selected opportunities, i.e. not computed only from a single opportunity or transition.
- On the first orbit, the start point is a virtual opportunity at time 0.0, without any transition constraint to any other opportunity, i.e. an edge exists from that virtual start vertex to every opportunity of the graph;
- On other orbits, the start point is the last selected opportunity for that platform (typically in the previous orbit, or before if none were selected in the previous orbit):
  - Note: it is possible for an opportunity A to start in an orbit and end in the next, and thus conflict time-wise with an early opportunity B in the next orbit. If opportunity A becomes the start vertex of the next orbit, it will not have edges to all other opportunities of the graph (i.e. not with B or other early opportunities).

4.1.1.3 Opportunity Selection Criterion

This criterion computes the value of a **candidate opportunity** for potential selection into a collection plan. It requires knowing:

- Current opportunity;
- Candidate opportunity (potential successor of current);
- Previously selected opportunities; and
- Data set where opportunities are initially from (for data on corresponding tasks, area overlaps, etc.).

Candidate opportunity value = added value / time delta

Where:

- **time delta** = candidate opportunity end time – current opportunity end time:
    - This includes platform idle time, transition duration, and candidate opportunity execution.
- **added value** = candidate *Opportunity differential objective value*:
    - *Opportunity differential objective* value is defined in section 2.1.2 Formulas; and
    - Roughly, this is the opportunity contribution to the whole collection value, given that opportunity and previously selected opportunities.

4.1.1.4 Output

Heuristic output is a collection plan. In essence, this is a list of selected opportunities (see 1.2 Collection Plan (CP) Data Model (1st draft).

4.1.2 Pseudo-code

solutionPlan ← ∅
For each platform **p** *do*
    current$_p$ ← nil
end For

For **orbit** *in* 1 ... (**horizon** / **orbitDuration**) *do*

    timeFrame ← **[orbit** * **orbitDuration**, (**orbit**+1) * **orbitDuration]**
    For each platform **p** do
        Create opportunity graph **G**$_p$:
            vertices ← **current**$_p$ ∪ **{**opportunities from **dataSet** for **p** and start within **timeFrame}**
            edges ← **{**time-wise possible transitions**}**
    end For

    While (**current**$_p$ has at least one successor for at least one platform **p**) *do*
        candidates ← ∅
        For each platform **p** *do*
            Remove from **G**$_p$ all *infeasible* outgoing edges from **current**$_p$
            candidates ← **candidates** ∪ **{**successors of **current**$_p$ in **G**$_p$**}**
        end For

If **candidates** ≠ ∅ *then*
For each opportunity **opp** *in* **candidates** *do*
criterion$_{opp}$ ← (re-)compute *Opportunity selection criterion* for **opp**
end For
Select opportunity **opp** *from* **candidates** that maximizes **criterion**$_{opp}$
solutionPlan ← **solutionPlan** ∪ **opp**
current$_{p}$ ← **opp**, where **p**: platform of **opp**.
end If
end While
end For

For each task **t** *do*
taskOpporunities ← {opportunities from **solutionPlan** for **t}**
If (*total coverage* for taskOpportunities) < **requiredTaskCoverage**$_{t}$ *then*
Remove each of **taskOpporunities** from **solutionPlan**
end If
end For

return **solutionPlan**

4.1.3 Input Data

This heuristic requires all data in the current implementation of the Collection Task Data Model (see section 1.1). As such the solver simply requires an interface to such data.

Given the lack of both real data and service interface for now, simulated data is used as described in section 6.1 Generated Random Input Data. Data is thus loaded from JSON or text files into an in-memory implementation of the data model, then fed to the solver.

4.1.4 Java Implementation

Implementation is split in three parts:

- Solver-specific code:
  - `DiscoverSolvers` module; and
  - `ca.gc.rddc.discover.solvers.mypicc` package.
- Opportunity graphs (common to different solvers):
  - `DiscoverSolvers` module;
  - `ca.gc.rddc.discover.solvers.common` package;
  - In its own class, `OpportunityGraph`;
  - Design notes:
    - To allow for the processing of very large graphs, the performance-oriented (speed and memory footprint) `grph` library was chosen to contain an underlying graph structure. See http://www.i3s.unice.fr/~hogie/software/index.php?name=grph.

- Calculations and data:
    - Common to different solvers and other modules; and
    - `DiscoverCommons` module.

Solver-specific code has only two classes:

- `MyPiccHeuristic`: the main solver class:
    - Other classes from the same package (`MyPiccGraphVisitor`. `MyPiccGraphVisitConstriants` and others), are actually inner workings of the heuristic that were broken down into those few classes to ease maintenance and unit testing. Those have package visibility and are not meant to be used outside the context of the heuristic without some refactoring.
- `MyPiccHeuristicListener`: a listener interface:
    - The solver classes support having listeners that are kept informed of the inner heuristic workings as it runs (updated selection criterion value, selected opportunities, rejected edges, etc.).

See source code (from the `DiscoverSolvers` and `DiscoverCommons` modules) and its documentation for more details.

4.1.5 Test and Validation

So far, the solver is part of a Java library and is not yet integrated within or to other services/applications. Therefore, there isn't any integration test yet. The solver is thus tested and validated through the following:

- Functional integrity/validation: automated unit tests (Junit framework);
- Performance/usability: manual test run with an in-house test console application (see section 6.5 Solver Launcher Console); and
- Manual validation: manual test runs with the in-house test console, augment some solver-specific output, and manual examination of that output.

4.1.5.1 Unit Tests

Running them requires a development environment.

In some tests, using an actual data set is more relevant than using mocked components. Those tests mostly use the same small-scale generated data set (see section 6.4.2 Small-scale Non-random Data Generation) as well as variants of it (i.e. data is modified in the unit test code only where such variants are to be tested). This data set is packaged as a JSON file resource along with the source code (and therefore available in the `DiscoverSolvers` module). Also, on top of their respective test-specific validations, those tests use a generic "solution feasibility test" to assert the solution validity and consistency.

4.1.5.2 Performance/usability

There is no precise benchmark/performance target for this solver, but the overall performance seems subjectively quite acceptable so far, given results in 4.1.6 Results.

4.1.5.3 Manual Validation

The manual validation was done through:

- Analyzing the reported collection plan on the console;
- Analyzing the "verbose" output of the solver:
  - Gathered via a solver *listener* implemented in the test console application.
- Looking at some generated opportunity graphs:
  - Gathered via a solver *listener* implemented in the test console application; and
  - Opportunity graphs also have vertices and edges colored/labeled according to intermediate results within the solver, i.e. they depict some of the inner workings. For example, here is this small graph from one of the platform/orbit of the medium set (most are much larger), with highlights on selected opportunities and rejected edges:

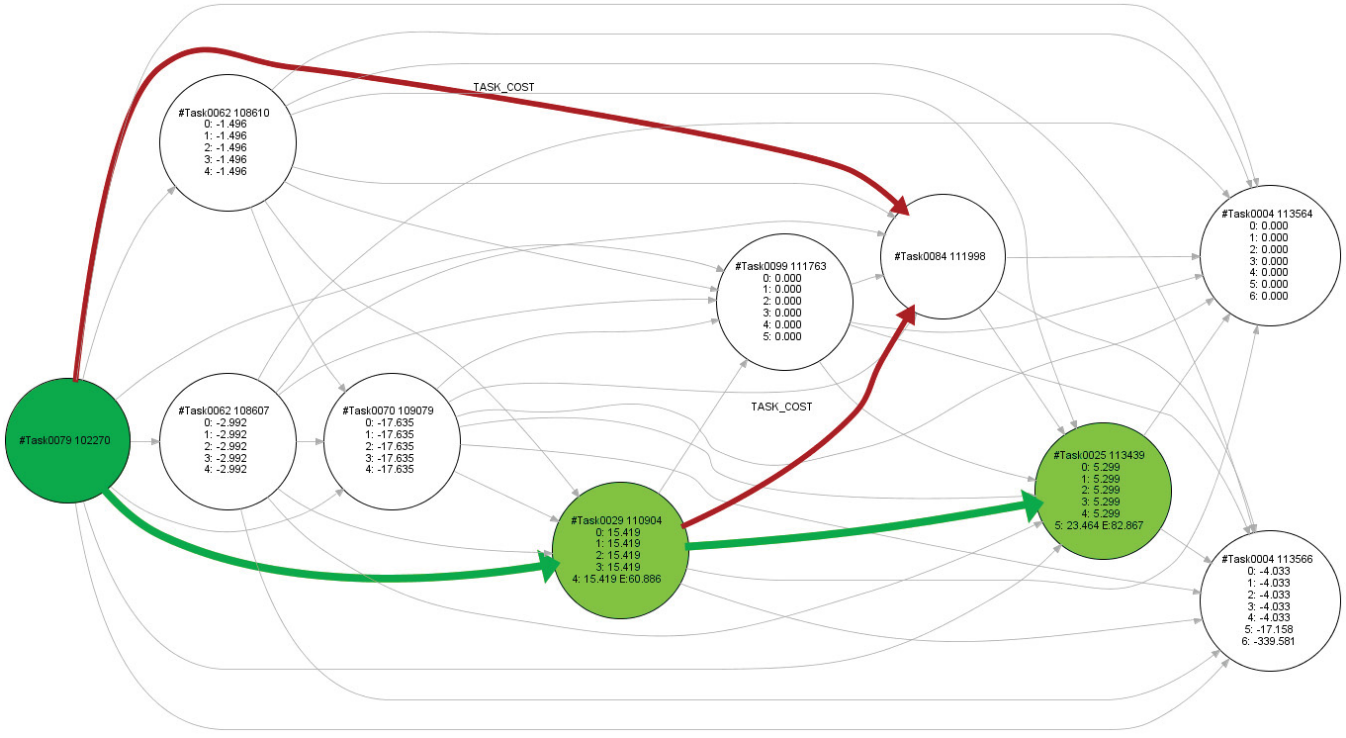


Figure 3: Example of MY-PICC Applied on Opportunity Graphs

- All the above was mostly done on medium-size set (the same as in the performance testing above) and the small-scale generated data set used for unit testing.

4.1.6 Results

Solver is tested (test bench and input data) with generated data as described in section 6 Simulation Data and Test bench. Using those data sets, results are:

Table 6: MY-PICC (Time-Weighted) Solver Results

Data Set	Result Collection Plan			Solver Run Time (seconds)
	Absolute Value	% of Upper Bound	Total cost (k$)	
Small (Wang)	6.668	75.3	N/A	0.006
Medium	48.14	71.3	463	0.162
Medium-large	153.1	72.9	1449	1.69
Large	541.7	77.3	4551	25.0

Results with Wang data sets are reported in section 7 Solvers Comparative Results.

How good does this solver perform in regard to a generated collection plan cannot be assessed from absolute results alone. Those probably depend on how much simulated data is fit or not for that particular solver and that does not tell exactly how it could behave with real-life data. It can be said, however, that collection value is significantly lower than GATHER and QUEST (on spot targets, large areas are not supported by QUEST) on any tested data set so far, although it runs significantly faster as well (and much more so).

4.2 MY-PICC - Plain-value variant

4.2.1 Description

This is exactly like the *Time-weighted variant* of MY-PICC, except that the opportunity selection criterion (see section 4.1.1.3 Opportunity Selection Criterion) is simply:

- candidate opportunity value = **added value.**

Instead of:

- candidate opportunity value = **added value / time delta.**

This difference is implemented as an option flag in `MyPiccHeuristic` class.

Everything else from the other heuristic (description, pseudo-code, input data, test and validation, etc.) still applies here.

4.2.2 Results

Solver is tested (test bench and input data) with generated data as described in section 6 Simulation Data and Test bench. Using those data sets, results are:

Table 7: MY-PICC (Plain-Value) Solver Results

Data Set	Result Collection Plan			Solver Run Time (seconds)
	Absolute Value	% of Upper Bound	Total Cost (k$)	
Small (Wang)	6.178	69.8	N/A	0.003
Medium	50.91	75.4	484	0.078
Medium-large	171.6	81.6	1611	1.62
Large	572.0	81.6	3816	22.3

As with the other MY-PICC variant, results with Wang data sets are reported in section 7 Solvers Comparative Results.

On generated data sets, results are slightly better in general than the Time-weighted variant of MY-PICC:

- Better collection value;
- Lower cost; and
- Faster run time.

On Wang data sets though, as presented in section 7 Solvers Comparative Results, collection value is significantly lower.

- On generated data sets, this can be explained by higher resource contention, especially on larger sets: PV can reach most-valued opportunities per orbit in most orbits, while TDW is likely to reach global limits earlier (task budget for instance) and thus be stuck with less-valued opportunities accepted in early orbits;

- On less constrained data set, such as Wang data sets, TDW fares better compared to PV. PV misses many opportunities and reaches the end of the time horizon while still far from constraints.

Obviously, those remarks hold only for the tested data sets, and may not be generalized to real-life data.

5 METAHEURISTICS MODELS/SOLVERS

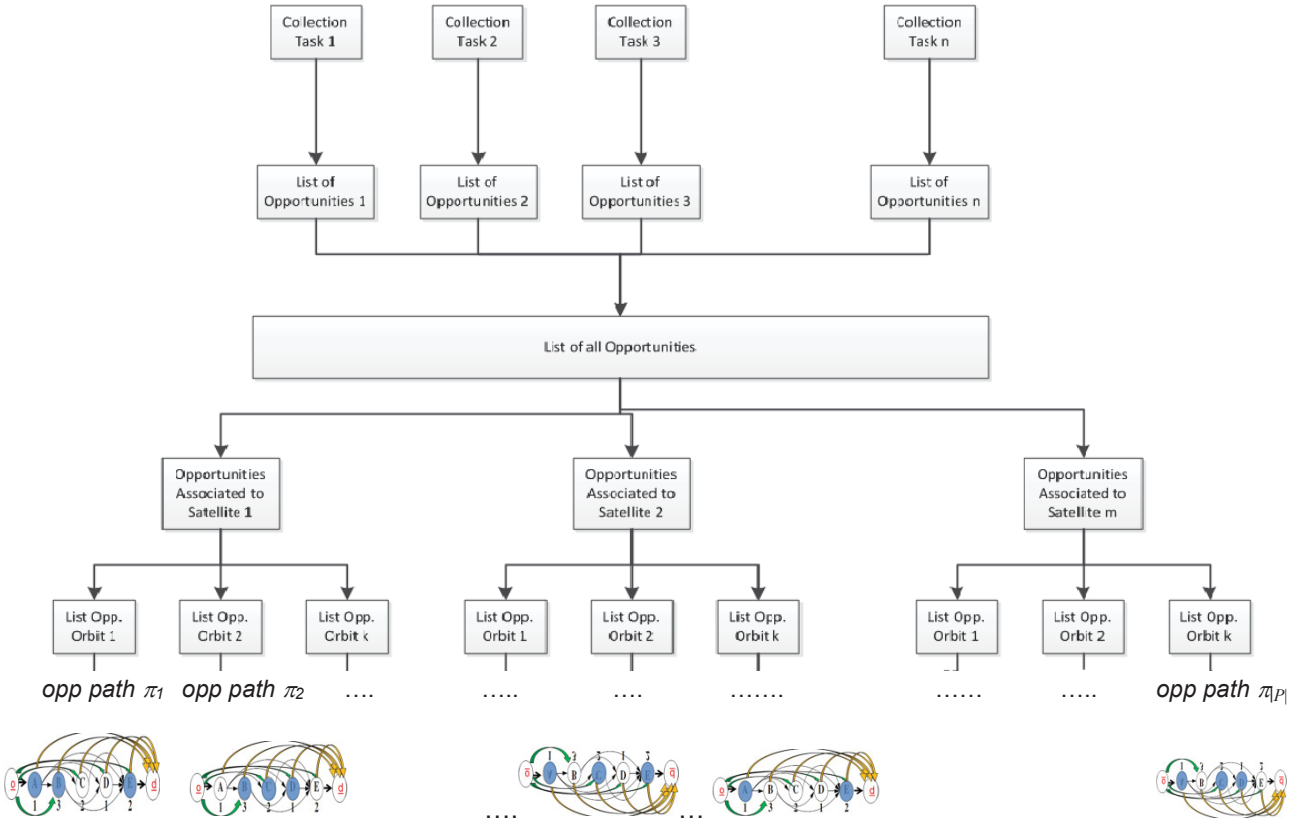
5.1 GATHER

5.1.1 Novelty

GATHER is a *Genetic Algorithm-based collecTion schedulER.*

Based on PMX variants extensions:

- Provides search guidance of the solution space using a satellite collection graph (satellite collection network flow) in new genetic operators to exploit natural opportunity (opp) precedence, which facilitates solution recombination and child production of timely collection opportunity sequences (time constraint violation reduction) in individual solutions:
    - The implicit high satellite collection (imaging opportunities) graph connectivity is exploited to efficiently repair temporal constraint violations.
- Maintain a minimum proportion of feasible (α) / unfeasible (1-α) sub-population solutions to better search/explore the solution space (escape local extrema);
- Exploit a serviceability matrix to crossover dissimilar orbits (from same or different satellites) sharing common serviceable tasks –dissimilar orbits selection proportional to a shared common task proportion;
- Mutator: change the collection opportunity sub-path using $G_\rho(V_\rho = O_\rho, A_\rho)$ graphs to easily replace a feasible sequence ($o_i \rightarrow o_j$, $o_j \rightarrow o_k$, $o_k \rightarrow o_l$, ….) and reconnect to the terminal collection opportunity path segments with minimal repair;
- Fitness: includes objective (Obj) and constraint violations; and
- Provides a better-quality solution than the best-known heuristic if the latter is used to generate initial population of feasible solutions (changing initial opportunity node to visit).



5.1.2 Definitions

In accordance with our data model, satellites are hereafter referred to as platforms.

5.1.2.1 Population

- A population individual is a *chromosome*, and inversely, the population is a set of such chromosomes;
- The population size is invariant over generations;
- Initial population individuals are generated from a genetic pool in the Genome (see 5.1.2.3 Genome); and
- On each generation, a minimal proportion α of the population individuals is repaired using the `ChromosomeSolutionRepair` mutator (see 5.1.3.2 Chromosome Solution Repair).

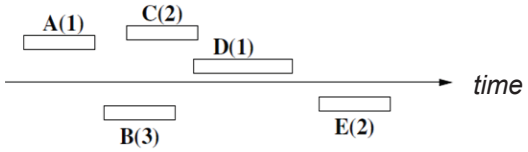
5.1.2.2 Individual

In its whole, a chromosome is a *multi-satellite collection schedule* solution, that is, a set of 'collection opportunity paths' across platform orbits for all platforms that can be put together in a collection plan.

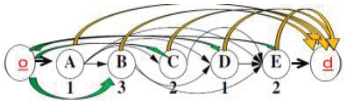
Chromosome composition:

- Each chromosome is made of a list of *alleles* (one on each *locus*, i.e. each position on a chromosome), each being tied to a *gene*;

- A **gene** *describes* a given locus on a chromosome:
    - In this solver, genes are tied to a single platform and a single orbit, collectively called ρ.in this section. For example: the gene at locus 13 is associated with orbit 3 of platform 1, for which a few opportunities are available:



    - Other ρ-related invariant data are pre-computed in the gene, such as the platform-orbit collection opportunity graph $G_\rho(V_\rho = O_\rho, A_\rho =$ legal chronological transitions), where O_ρ imposes a partial natural order on precedence (based on the opportunity start time) on its elements:



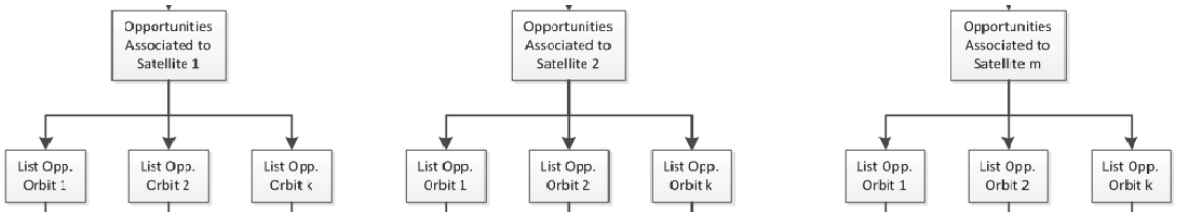
    - Genes are stored in the genome, and only referenced to by alleles and chromosomes.
- An **allele** is the *current value* (genetic code) of a gene on a chromosome:
    - In this solver, alleles represent a chronologically feasible (and potentially empty) set of opportunities, i.e. an *opportunity path* π in the opportunity graph G_ρ of the corresponding gene:

$$o_{i\rho} \in \pi_\rho \subseteq O_\rho; \quad (o_{i\rho}, o_{j\rho}) \in A_\rho = \{a_{ij\rho}\}, a_{ij\rho} \in \{0,1\} \quad i < j$$

$$\pi_\rho = o_{\pi(1)\rho}, o_{\pi(2)\rho}, \ldots o_{\pi(t)\rho}, o_{\pi(t+1)\rho}, \ldots o_{\pi(|\pi_\rho|)\rho} \quad \pi(t) < \pi(t+1)\,[timeline]\; t \in \{1..|\rho|\}\; from\; G_\rho(V_\rho = V_\rho, A_\rho)$$

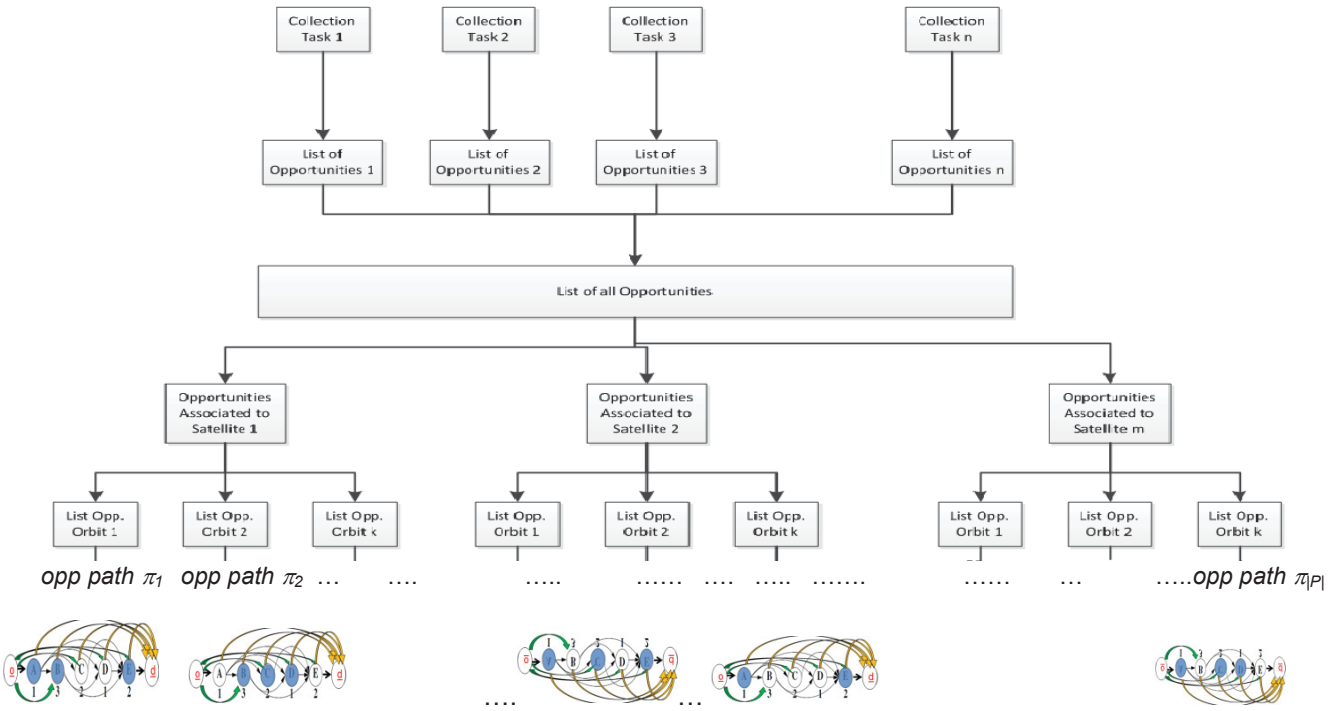
    - For example, alleles on locus 13 could contain opportunities 45, 17 and 69 (from the global list of all opportunities), all of which (in that order) form a valid path in the opportunity graph of genes on locus 13.
- Genes in genome and alleles in chromosomes are sorted by platform then by orbit, as illustrated below:

$$\pi_1 | \pi_2 | \ldots | \pi_\rho | \ldots | \pi_{|P|} \quad where\; \pi_\rho : opp\; path\; in\; G_\rho(V_\rho = V_\rho, A_\rho)$$



2	5	23

$$\pi_\rho = o_{\pi(1)\,\rho}, o_{\pi(2)\,\rho}, \ldots o_{\pi(t)\,\rho}, o_{\pi(t+1)\,\rho}, \ldots o_{\pi(|\pi_\rho|)\,\rho} \quad \pi(t) < \pi(t+1), a_{o_{\pi(t)}\,o_{\pi(t+1)}\,\rho} = 1, \textit{from } G_\rho(V_\rho = V_\rho, A_\rho)$$



Chronological feasibility:

- On a given platform, an opportunity must start in time after the platform is ready for it (according to the previous opportunity end time, transition and set-up delays, etc.);
- Applies locally (within each allele):
  - Imposed by allele definition.
- Applies globally (between alleles):
  - Chronological criteria must hold between the last opportunity of an allele and the first of the next allele, only if those alleles are from consecutive orbits on the same platform; and
  - Imposed by a mandatory chronological repair in genetic operators.
- In a population, chromosomes are always chronologically feasible.

Solution feasibility:

- Despite chromosomes being chronologically feasible in the population, a chromosome may nevertheless be *infeasible* as a collection plan solution
  - It may fail meet various data set constraints; and

- See 1.1.7 Constraints set, for global, platform and per-orbit constraints.

- Chromosomes can be repaired for such feasibility, as described in 5.1.3.2 Chromosome Solution Repair.

5.1.2.3 Genome

The genome is the genetic pool. It contains all genes, each of them describing alleles in chromosomes. Also, for each gene, the genome contains a list of *valid* alleles.

Allele validity is defined as:

- Chronologically feasible; and

- Solution-feasible on local scale (this allele only), but not once it joins others on a chromosome:

  - That is, if it were to be the only allele on a chromosome, the chromosome would be solution-feasible, as defined in 5.1.2.2 Individual.

The genome used in this algorithm is not *complete*:

- It lists several valid alleles for each gene, but not *all* possible alleles (obvious combinatory issue); and

- Typically used to generate an initial population with a decent genetic code, or by some mutators to replace an allele with a valid one later on when evolving the population.

The list of valid alleles for each gene is created using the heuristic method (see 4.1 MY-PICC - Time-weighted variant) applied on a platform-orbit opportunity graph independently. This heuristic is deterministic and thus only generates a single possible path for a given graph. Therefore, various opportunity nodes in the graph are used as the visit starting point to generate various alleles.

5.1.2.4 Fitness

Given that a chromosome has a one-to-one mapping to a collection plan, part of it fitness value can be evaluated using the objective function (see 2 Objective Function):

Fitness:

= *bonus* (collection value) – *malus* (constraint violation penalty)

= collection value – *λ* * *constraints violation proportion*

- Where the *constraints violation proportion* is the ratio of counted constraint violations over the possible total number of potential violations; and

- Where *λ* is set to the plan objective value upper bound to balance the relative weights of the bonus and malus.

$$= Obj_{Plan} \quad - \quad \sup Obj_{Plan} \sum_{\rho} \sum_{\substack{RES=\{energy,\\ memory,time\}}} \frac{1}{2}\left(1+\tanh\left(\overbrace{M}^{10^5}\left(RES_{\pi_\rho}-RES_{\rho}\right)\right)\right) \Bigg/ \sum_{\rho} \sum_{\substack{RES=\{energy,\\ memory,time\}}} 1$$

Given a large enough *M* value, the above *tanh* is used to approximate an on/off constraint violation:

$$\frac{1}{2}\left(1+\tanh\left(\overbrace{M}^{10^5}\left(RES_{\pi_\rho}-RES_\rho\right)\right)\right)\approx\begin{cases}1 & if\ RES_{\pi_\rho}>RES_\rho\\ 0 & otherwise\end{cases}$$

Note: spot targets ($A_{rso\rho}/A_{rs}$ =1) multi-visits are implicitly discouraged by the objective function.

5.1.2.5 Serviceability

Given a task *r* and an orbit ρ, the serviceability matrix is defined as:

$$svc(r,\rho)=\begin{cases}1 & if\ task\ r\ serviceable\ over\ orbit\ \rho\\ 0 & otherwise\end{cases}$$

A task is serviceable if at least one opportunity in an orbit covers that task. Also, for the sake of simplicity, we define orbit ρ as an orbit on a specific platform, i.e. orbits from different platforms occurring at the same time are attributed a different ρ. In other words, ρ is akin to the locus in the genome and chromosome.

5.1.3 Generations

Key points:

- Children pair generation:
    - Two parents produce two children, initially clones of both parents:
        - Parents are randomly picked according to a rank-based probability. See 5.1.3.1 Parent Selection.
    - Children undergo a sequence of zero to many genetic transformations through genetic operators:
        - Genetic operators to apply are chosen randomly;
        - Each available operator can be applied at most once, with some probability to be applied;
        - Operators application order is random; and
        - Operators are chained: an operator output (two children) is taken as the next operator output (two parents).
    - All operators are responsible to repair children for *chronological* feasibility (not *solution* feasibility) on output, as described in 5.1.2.2 Individual:
        - The chronological repair is performed both within alleles and between consecutive alleles on the same platform;
        - Problematic opportunities are removed from alleles until the whole chromosome is chronologically sound; and

- When an opportunity needs to be removed in a pair of conflicting opportunity, either one is chosen in a probabilistically random fashion. Each is attributed a probability proportional to its *single opportunity value*, and the randomly picked opportunity is *kept* (other is removed). That is, the least-valued opportunity is more likely to me removed.

- Elitism is applied to keep a portion of the fittest individuals from the previous generation. The remaining of the population is completed with the fittest individuals from the new generation;

- Algorithm stops when one of the following conditions is met (all of those are verified simultaneously):
  - The value of the fittest individual (best computed *fitness value*) did not improve *significantly* (as defined below) over the last *n* generations:
    - A simple linear regression is performed over the fittest individual fitness value from each of the last *n* generations. Only generations where the best fitness value is positive are used (normally always the case, except for theoretical corner cases);
    - Generations are mapped to the x-axis: [*n-1* generations earlier, latest generation] is mapped to [0, 1];
    - Fitness values are mapped to the y-axis: [0, max fitness] is mapped to [0, 1]; and
    - Stop condition is met when the regression slope is smaller than a specified threshold.
    - Both the slope threshold and the number of generations are specified as solver parameters.
  - Failure to generate individuals with positive fitness value over the first *n* generations (theoretical corner case, very unlikely in reality); or
  - Hard-limit on generation count.

5.1.3.1 Parent Selection

A rank-based fitness scaling scheme inspired by Potvin & Bengio (The Vehicle Routing Problem with Time Windows Part II: Genetic Search) is used (fitness-based parent selection strategy, etc.). In the current implementation, the last paragraph in the above approach is not applied.

In order to bias the selection process towards the best solutions, a linear ranking scheme is used.[13] First, all solutions in the current population are ranked according to their quality, that is, the best solution has rank 1, and the worst solution has rank P, where P is the size of the population. A fitness value is associated to the solution of rank i as follows:

$$\text{fitness}_i = \text{MAX} - [(\text{MAX} - \text{MIN}) \times (i-1)/(P-1)]$$

$$\text{with } \text{MAX} = 1.6, \text{MIN} = 0.4.$$

For example, when P=5, $\text{fitness}_1=1.6$, $\text{fitness}_2=1.3$, $\text{fitness}_3=1.0$, $\text{fitness}_4=0.7$ and $\text{fitness}_5=0.4$. Then, a fitness-proportional selection scheme is applied to these values. Namely, the selection probability p_i for a solution of rank i is:

$$p_i = \frac{\text{fitness}_i}{\Sigma_{j=1,\ldots,P}\ \text{fitness}_j} = \frac{\text{fitness}_i}{P} .$$

It is worth noting that the summation over all fitness values in the population is equal to P, because MIN + MAX = 2, and the average fitness is equal to 1. Since P different selections (with replacement) must be performed on the current population in order to select P parents and generate P new offspring, the expected number of selections E_i for a solution of rank i is:

$$E_i = P \times p_i = \text{fitness}_i .$$

Hence, fitness_i is also equal to the expected number of selections for the solution of rank i. For example, the best solution with fitness MAX=1.6, is expected to be selected 1.6 times on average over P different trials.

In order to reduce the variance associated with pure proportional selection (i.e., each solution can be selected between 0 and P times over P different trials), stochastic universal selection or SUS was applied to the fitness values. This approach guarantees that the number of selections for a given solution is at least the floor, and at most the ceiling, of its expected number of selections.[1]

5.1.3.2 Chromosome Solution Repair

For a chromosome to represent a feasible solution, it must observe both implicit physical time constraints (chronological feasibility) and explicit constraints from the data set. While *chronological* feasibility is mandatory and enforced as generated children go through genetic operators, *solution* feasibility is mandatory only for the final single returned solution. From a generation to another, the population contains both solution-feasible and solution-infeasible chromosomes to have both viable solutions and a diverse genetic pool.

Solution feasibility is defined by satisfying various constraints, essentially listed in the data model (see section 2.1.7 Constraints set). Within a chromosome, constraint violations occur either when:

- Exceeding capacities:
    - Orbital platform capacity: energy, memory, thermal;
    - Task capacities: cost; and
    - Overall capacity: total budget.

- Not meeting minimum requirements:
    - Minimal task coverage; and
    - Other requirements, not yet implemented in this algorithm:
        - Mandatory task visits;
        - Precedence task constraints; and
        - Stereo/concurrent observation/cross-cueing/change-anomaly detection.

While exceeded capacities can be fixed by removing some opportunities from alleles, unmet requirements would be met by adding opportunities. Choosing which one to add/remove is not trivial. Furthermore, opportunities may need to be removed to fix an artificial paradox: for a given task, as the number of opportunities covering that task *increases*, the task objective value (and area coverage) may numerically *decrease*. This paradox makes minimum requirement evaluation unreliable and hinders the ability to select proper opportunities to remove to fix exceeded capacities.

Therefore, the solution repair process has to be divided into a few steps:

- Address the coverage and objective value calculation paradox;
- Fix exceeded orbital/task constraints;
- Fix unmet minimum requirements; and
- Fix exceeded global constraints.

Coverage and Objective Value Calculation Paradox

In reality, adding an opportunity may either increase the total coverage or at worst not bring anything new (if the area covered by that opportunity is already fully covered by others). The same goes for task and collection value. However, the current coverage calculations, derived from the objective function considers only opportunity individual coverage and pairwise overlaps, but not three-way or more overlaps. See section 2.1.2 Formulas for task value and relative area coverage. As such, an opportunity for which the sum of pairwise overlaps with other planned opportunities is greater than its own coverage will decrease the overall coverage. The same occurs to task objective value given reasonable p and q values.

To fix this paradox, some opportunities need to be removed until remaining opportunities have an overall positive contribution to the task value and coverage. There is no way to know which opportunity effectively brings a positive or negative contribution since this contribution depends on other opportunities currently selected (opportunity value and overlap values). Therefore, all opportunities currently selected in a chromosome have their respective *Opportunity differential objective value* computed (see section 2.1.2 Formulas as well). This provides at least a hint on opportunities that are more likely than others to bring a positive/negative contribution.

The opportunity with the worst contribution is removed, then contributions of remaining opportunities are recomputed, and the process repeats. It stops when removing the worst opportunity would not increase task objective value anymore. It can also stop earlier, if current task objective value (using remaining opportunities) is greater or equal to the best *single opportunity value*. In other words, although task value could perhaps be improved further by removing opportunities, reaching the best single opportunity value is considered fair enough, and helps preserve some genetic diversity.

Note that such paradox may not exist anymore in a future version of this software, if:

- Opportunity coverage is expressed in actual area geographic coordinates instead of in a simple area coverage relative to a task area of interest; and
- Collection value is computed accordingly.

As a side effect, fixing this paradox also effectively reduces constraints violations and reduces the need to delete opportunities in the next steps below.

Fix exceeded orbital and task constraints

Orbital capacity issues are resolved by removing opportunities locally in each allele (i.e. independently on each platform/orbit pair). Opportunities to remove are selected randomly but with a relative probability equal to:

1 - *single opportunity value* × *task nominal value*

In other words, opportunities that probably contribute less individually (actual overall contribution also depends on opportunity overlaps, which is not computed here) are more likely to be removed. Process is repeated until all orbital constraints are met. Removing them randomly-yet-probabilistically instead of deterministically preserves genetic diversity while aiming at maximizing collection value.

Task constraints (only maximum budget for now) are dealt with after orbital constraints, since meeting them involves removing opportunities across the whole chromosome, on a per-task basis. Obviously, it may happen that orbital repairs already fixed task constraints as a side effect. Opportunities to remove are selected in a similar fashion that with orbital constraints, but across the whole chromosome instead of within a specific allele.

Fix Unmet Minimum Requirements

Obviously, fixing such requirements by adding tasks would compete with previous repairs. Therefore, unmet task requirements are rather fixed by removing all opportunities related to offending tasks, effectively rejecting entire tasks from the solution. The improvement of the overall collection value thus relies on the whole genetic algorithm to bring back in some chromosomes, in a later generation, enough opportunities to meet minimum requirements, if even possible.

Fix Exceeded Global Constraints

At this point, only the overall budget constraints remain to be satisfied. Again, to avoid breaking previous repairs, the simplest thing to do is to remove complete tasks, starting with the least valued/most expensive tasks. That is, tasks, along with their currently selected opportunities, are sorted according to:

task objective value / task cost

Tasks with the lowest ratio are removed until budget constraints are met. A more advanced heuristic could cherry pick opportunities across tasks that can be removed without breaking again any previous repair, but this isn't implemented yet.

5.1.4 Genetic operators

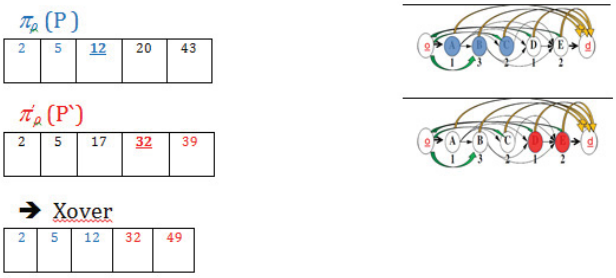
5.1.4.1 Recombination – Same Orbit X_{SO}

Same orbit collection opportunity path recombination between two parents.

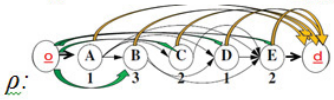
Main operations:

- Select random orbit ρ in both P in P' parents, with their opportunity paths π_ρ and π'_ρ in their respective corresponding allele;
- Select crossover opportunity point o_ρ (P) from P; and
- Find the earliest $o'_\rho(P':\pi'_\rho) > o_\rho(P: \pi_\rho)$ and swap sub-paths to generate a child orbit π^X_ρ respecting opportunity transition constraints (start time precedence) – legal repair using valid transition from A_ρ if required.

$X_{SO}(\ P(\pi_1, \pi_2 \ldots \boldsymbol{\pi_\rho} \ldots \pi_{|P|})\ X\ P'(\pi'_1, \pi'_2 \ldots \boldsymbol{\pi'_\rho} \ldots \pi'_{|P|})\ ) \rightarrow P^c\ (\pi_1, \pi_2 \ldots \boldsymbol{\pi^X_\rho} \ldots \pi_{|P|})\ ,\ P^{c'}(\pi'_1, \pi'_2 \ldots \boldsymbol{\pi'^X_\rho} \ldots \pi'_{|P|})$



*Single child generation to reinforce opp_temporal ordering.



P1(ABCE) x P2(BCD) →Child(ABD) (Child(BDE) not necessarily feasible)

5.1.4.2 Recombination - Orbit Swap X_{OS}

This operator exchanges 'full orbit collection opportunity path' solution between corresponding similar platforms from 2 parent solutions P and P', i.e. it exchanges corresponding alleles between two chromosomes: $\boldsymbol{\pi_\rho \Leftrightarrow \pi'_\rho}$

$X_{OS}(\ P(\pi_1, \pi_2, \ldots \boldsymbol{\pi_\rho}\ldots, \pi_{|P|})\ X\ P'(\pi'_1, \pi'_2, \ldots \boldsymbol{\pi'_\rho}\ldots, \pi'_{|P|})\ ) \rightarrow P^c\ (\pi_1, \pi_2, \ldots \boldsymbol{\pi'_\rho}\ldots, \pi_{|P|})\ ,\ P^{c'}(\pi'_1, \pi'_2, \ldots \boldsymbol{\pi_\rho}\ldots, \pi'_{|P|})$

5.1.4.3 Mutation – Cross-Orbit X_{CO}

This mutator aims to reduce the same task coverage across different orbits. It is applied independently on each parent to produce offspring, i.e. it does not mix genetic content between them.

Main operations:

- Select two dissimilar orbits ρ,ρ' sharing common tasks among their potential opportunities (from all possible opportunities in the genome, not only those from current alleles in parents):
  - Select pair at random but give higher priority to pairs of orbits sharing more serviceable tasks, i.e. proportional to $|O_{\rho\rho'}|$ where:
    - $O_{\rho\rho'}$: set of shared serviceable task r such that $svc(r,\rho) * svc(r,\rho') = 1$.

- Probability of selecting ρ,ρ' from all possible pairs is proportional to $\left|O_{\rho\rho'}\right|$.

- Remove potentially shared task opportunities from either orbit with the hope that they might pop back in the other in the last step:
  - In π_ρ, remove opportunities for shared tasks that are not found in $\pi_{\rho'}$, and inversely for $\pi_{\rho'}$ and π_ρ.; and
  - For opportunities of shared tasks found in both orbits, remove either one probabilistically, based on *single opportunity value*.

- Add as many opportunities as possible in both π_ρ and $\pi_{\rho'}$ independently as follows:
  - Consider all opportunities not already in π and that were not previously removed;
  - Sort them by 'value' (most valued first); and
  - Try to add them one by one, but reject those that can't fit the current graph when added.

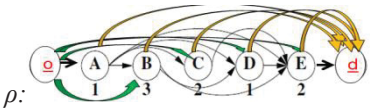
5.1.4.4 Mutation – Sub-Path

This operator is applied independently on each parent to produce offspring, i.e. it does not mix genetic content between them. In a random allele, it removes a part of it (a collection opportunity sub-path π_ρ) and bridges remaining segments with time-wise possible opportunities from the opportunity graph.

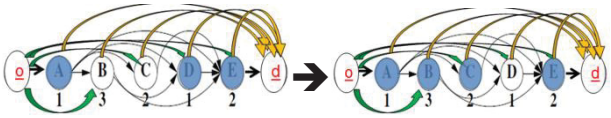
$$P\ (\pi_1, \pi_2 \ldots \boldsymbol{\pi_\rho} \ldots \pi_{|P|}) \rightarrow \mu(P) \rightarrow P^\mu\ (\pi_1, \pi_2 \ldots \boldsymbol{\pi^\mu_\rho} \ldots \pi_{|P|})$$

$$\pi_\rho = o_{\pi(1)\rho}, o_{\pi(2)\rho}, \ldots . \overline{\left[o_{\pi(t)\rho}, o_{\pi(t+1)\rho}\right]} \ldots . o_{\pi(|\pi_\rho|)\rho} \rightarrow$$

$$\pi^\mu_\rho = o_{\pi(1)\rho}, o_{\pi(2)\rho}, \ldots . o_{\pi(t-1)\rho}, \overline{\left[o'_{\pi(t)\rho}, \ldots, o'_{\pi(t+k-1)\rho}\right]}, o_{\pi(t+k)\rho} \ldots . o_{\pi(|\pi_\rho|)\rho}$$



π_ρ: 'A**D**E' → π^μ_ρ: 'A<u>BC</u>E'



The sequence to remove is determined from a contention-based random start point and a fully random length:

- Each opportunity in the allele refers to a task (target) *r*, typically a different task for each opportunity;
- Each opportunity is given a relative probability to be chosen, proportional to contention over *r* (C_r):
  - $p_r = \frac{C_r}{\sum_{r \in \pi_\rho} C_r} \qquad C_r = \sum_{\rho \in \mathrm{P}} svc(r,\rho) \quad svc(r,\rho) = \begin{cases} 1 & if\ r\ serviceable\ over\ \rho \\ 0 & otherwise \end{cases}$
- The higher that relative probability, the higher the odds of selecting that opportunity as the sequence start point; and
- Sequence length is uniformly random over the length of the allele segment that follows that start point.

Replacement opportunities are selected randomly from time-wise possible opportunities for that orbit, favouring opportunities with a low task serviceability over the complete time horizon:

- The two remaining allele segments (before and after the removed sequence), each of which is potentially empty, define the time bounds for replacement opportunities:
  - Orbit time bounds are used for empty segments.
- From the opportunity graph vertices, each opportunity within time bounds is a replacement candidate, regardless of existing edges between vertices;
- Candidate opportunities are added individually, each with probability $1 - p_r$, reusing p_r computed above; and
- A chronological repair (see 5.1.3 Generations) is applied on the allele to remove, if needed, opportunities that violate time constraints.

5.1.4.5 Mutation – Full Path

This operator is applied independently on each parent to produce offspring, i.e. it does not mix genetic content between them. It selects a random gene then replaces its whole allele with a randomly selected feasible allele from the gene allele pool (same pool that is used to generate the initial population, see 5.1.2.3 Genome).

5.1.5 Pseudo-code

5.1.5.1 Initial population

Once the genome is created (as described in section 5.1.2.3), the initial population can be created:

population ← ∅
For 1 ... **populationCount** *do*
    chromosome ← ∅
        For **locus** in 0 ... **geneCount**-1 *do*
        allelePool ← **{**possible alleles for gene **# locus** from **genome}**
        chromosome ← **chromosome** ∪ random allele from **allelePool**
    end For
    compute fitness for **chromosome**
    population ← **population** ∪ **chromosome**

end For
Sort **population** *on* decreasing chromosome fitness

5.1.5.2 Genetic evolution loop

Once the initial population is created, the main loop of the genetic algorithm kicks in to generate the solution collection plan:

While stop condition is not met, *do*
    children ← ∅
    For 1 ... **populationCount**/2 *do*
        parent1, parent2 ← two ranked-based probabilistic random elements from **population**

        operatorsToApply ← **{**probabilistic random selection from **allAvailableOperators}**
        Shuffle **operatorsToApply**

        child1 ← replicate **parent1**
        child2 ← replicate **parent2**
        For each **operator** in **operatorsToApply** *do*
            Apply **operator** *to* **(child1, child2)**
        end For

        compute fitness for **child1**
        compute fitness for **child2**

        children ← **children** ∪ **{child1, child2}**
    end For

    Sort **children** *on* decreasing chromosome fitness
    eliteCount ← **elitismRatio * populationCount**
    newPopulation ← **{eliteCount** first elements of **population}**
                ∪ **{(populationCount – eliteCount)** first elements of **children}**

    toRepair ← **repairRatio * populationCount**
    For each **chromosome** in **{toRepair** random elements from **newPopulation}** *do*
        Repair **chromosome** for *partial* solution feasibility
        compute fitness for **chromosome**
    end For

    Sort **newPopulation** *on* decreasing chromosome fitness
    population ← **newPopulation**
end While

For each **chromosome** in **population** *do*
    Repair **chromosome** for *full* solution feasibility
end For
Sort **population** *on* decreasing chromosome fitness
bestChromosome ← first element of **population**
solutionPlan ← **{**all opportunities from **bestChromosome}**

5.1.6 Input data

This heuristic requires all data in the current implementation of the Collection Task Data Model. See section 1.1. As such the solver simply requires an interface to such data.

Given the lack of both real data and service interface for now, simulated data is used as described in section 6.1 Generated Random Input Data. Data is thus loaded from a JSON file into an in-memory implementation of the data model, then fed to the solver.

5.1.7 Java Implementation

Implementation is split in three parts:

- Solver-specific code:
  - `DiscoverSolvers` module;
  - `ca.gc.rddc.discover.solvers.gather` package; and
  - `ca.gc.rddc.discover.solvers.gather.geneticoperators` package.
- Opportunity graphs:
  - Common to different solvers;
  - `DiscoverSolvers` module;
  - `ca.gc.rddc.discover.solvers.common` package;
  - In its own class, `OpportunityGraph`; and
  - Design notes:
    - To allow for the processing of very large graphs, the performance-oriented (speed and memory footprint) `grph` library was chosen to contain an underlying graph structure. See http://www.i3s.unice.fr/~hogie/software/index.php?name=grph.
- Calculations and data model:
  - Common to different solvers and other modules; and
  - `DiscoverCommons` module.

Solver-specific code if further split into several classes:

- `ca.gc.rddc.discover.solvers.gather` package:
  - `GatherGeneticAlgorith`: the main algorithm class;
  - `GatherGeneticAlgorith`: a listener interface:
    - The main solver class supports having listeners that are informed of the inner workings as it runs (chosen parents, created children, etc.); and

        - This was only barely exploited so far in the development. Might be removed in a future version if not really used.

    - Various functional components of the algorithm extracted as separated classes (`ChromosomeSolutionRepair`, `SingleOpportunityEvaluator`, etc.); and

    - Genetic data classes (`Chromosome`, `Gene`, etc.).

- `ca.gc.rddc.discover.solvers.gather.geneticoperators` package:

    - One class per operator, such as `MutatorSubPath`; and

    - Other helper classes used by operators only.

While most concepts explained in this document translates to closely equivalent code (such as `List<Opportunity>` to represent opportunity graph sub-paths in alleles), some mathematical concepts are not as much straightforwardly translated. For example, the serviceability matrix is implemented as a list of serviceable tasks in each `Gene` instance, for both convenience and performance reasons.

See the source code (from the `DiscoverSolvers` and `DiscoverCommons` modules) and its documentation for more details.

5.1.8 Test and Validation

So far, the solver is part of a Java library and is not yet integrated within or to other services/applications. Therefore, there isn't any integration test yet. The solver is thus tested and validated through the following:

- Functional integrity/validation: automated unit tests (Junit framework); and

- Performance/usability: manual test run with an in-house test console application (see section 6.5 Solver Launcher Console).

5.1.8.1 Unit tests

Running them requires a development environment.

In some tests, using an actual data set is more relevant than using mocked components. Those tests mostly use the same small-scale generated data set (see section 6.4.2 Small-scale Non-random Data Generation) as well as variants of it (i.e. data is modified in unit test code only where such variants are to be tested). This data set is packaged as a JSON file resource along with the source code (and is therefore available in the `DiscoverSolvers` module). Also, on top of their respective test-specific validations, those tests use a generic "solution feasibility test" to assert solution validity and consistency.

5.1.8.2 Performance/Usability

There is no benchmark/performance target for this solver. Overall performance seems acceptable in most cases (less than one minute), but might not be very usable with large data sets (several minutes) in an interactive scenario. It is worth nothing that execution times may largely vary according to algorithm parameters, but improving the run time that way also adversely affects resulting collection plan value. See results and discussion in 5.1.9 Results.

5.1.9 Results

5.1.9.1 Algorithm parameters

Several parameters in this solver may largely affect results. Those were reasonably tuned for various test data, but no fine tuning was performed for real-life data. Indeed, currently used data sets are generated and may not be representative of real-life data. Nevertheless, a “slightly better but much slower” configuration variant for GATHER was used with Wang data sets (in section 7 Solvers Comparative Results), and parameters were fine-tuned for that case.

By default, values are set to commonly used values in genetic algorithms, with minimal tweaks in cases that needed it most. Default parameters values are:

- Population size:
    - 30.
- Parent-picking ranking values (highest to lowest ranking value ratio):
    - 4.0; and
    - This generates ranking values from 0.4 to 1.6, as described in 5.1.3.1 Parent Selection.
- Individual probability for each genetic operator:
    - Recombination – Same Orbit: 0.50;
    - Recombination – Orbit Swap: 0.50;
    - Mutation – Cross-Orbit: 0.50;
    - Mutation – Sub-Path: 0.25; and
    - Mutation – Full Path: 0.25.
- Minimum solution repair ratio:
    - 0.50; and
    - The aim was to use a fairly low value, such as 0.25 to preserve genetic diversity. However, at first glance, results are somewhat poorer on the largest data set with low values (this was not investigated at this stage). On the contrary, a higher value does not seem to affect (neither for better nor worse) results on other data sets. Overall, this repair has a very high incidence on the result, as discussed in 5.1.9.4 Chromosome repair influence.
- Elitism ratio:
    - 0.07, i.e. 2 individuals in the current population.

- Stop conditions:
    - Maximum number of generations to allow near-identical fitness for the fittest individual:
        - 50; and
        - Lower value reduces run time, but also hinders improvements over heuristic solvers that this algorithm tries to leverage.
    - Fitness difference threshold for above 'near-identical' fitness:
        - 0.1%; and
        - Higher threshold reduces run time, but reduces result values as with the previous parameter.
    - Hard limit on generation count:
        - Unlimited.

5.1.9.2 Slower variant for Wang data

GATHER was used with an alternate configuration on the complete Wang data to see how close this non-deterministic algorithm could get to QUEST, albeit with longer running times (see section 7 Solvers Comparative Results).

Default parameters were reused, except for:

- Population size:
    - 150: this maintains a fuller genetic pool without having to rely solely on some mutators to bring back by chance some less-valued genetic material (opportunities). Although some opportunities can have a lower value when compared to other in specific cases (such as opportunity conflicts regarding time or other constraints), they might bring better value overall when selected with and without some others;
- Minimum solution repair ratio:
    - 0.65: for some reason, this value yields better output collection value;
- Stop conditions:
    - Maximum number of generations to allow near-identical fitness for the fittest individual:
        - 500: wait longer to make sure population is really not improving much;
    - Fitness difference threshold for above 'near-identical' fitness:
        - 0.01%: 10 times stricter than default value.

5.1.9.3 Execution results on generated data

Given the random nature of this algorithm, the solver is run 5 times on each data set, and results are averaged in Table 8 below. Nonetheless, minimum and maximum returned collection plan values are reported to show that variance is quite low on that regard, except for the largest data set. Also, GATHER givers better results than MY-PICC, except again for the largest data set. Causes for such discrepancies were not investigated at this stage.

Results on Wang data sets are found in section 7 Solvers Comparative Results, along with results from other solvers.

Table 8: GATHER Solver Results

Data Set	Result Collection Plan					Average Solver Run Time (seconds)
	Absolute Value			Average % of Upper Bound	Average Total Cost (k$)	
	Min	Max	Average			
Small (Wang)	7.245	7.245	7.245	81.8	N/A	0.204
Medium	59.47	59.98	59.69	88.4	509	3.41
Medium-large	192.9	198.3	195.5	93.1	1539	27.5
Large	533.2	566.2	549.8	78.5	3404	205

5.1.9.4 Chromosome repair influence

Given the high task opportunities contention over limited resources (satellites and their constraints), especially in the largest set, the chromosome solution repair mutator seems to have a significant influence on the result.

Indeed, once repaired, chromosome fitness increases significantly, making repaired chromosomes more and more likely to be chosen as parents for the next generation. Within a few generations only, those repairs propagate through the population more than anything else. Most of its influence is thus from how it is implemented (for example, fixing the “more is less paradox”, or opportunity selection criteria for removal).

Given this repair is so strong, it could potentially defeat other benefits of using genetic algorithms. Further investigation is warranted to evaluate how much imbalance it actually brings and what could be done against it if needed.

6 SIMULATION DATA AND TEST BENCH

Given the lack of real data (except for some foreign data that follow a simpler data model [1]), most solvers are developed and tested with generated data. Data is either:

- Generated through an in-house console application (described in section 6.4 Data generator application):
    - Used by MY-PICC and GATHER;
    - Once generated, the same sets are reused for testing; and
    - See section 6.1 Generated Random Input Data.
- From the author of [1], hereafter called *Wang data*:
    - Used by MY-PICC, GATHER and QUEST; and
    - See section 6.2 Wang data sets.

Also, since solvers are not yet integrated into a back-end service, they are only provided within a Java library. As such, solvers are launched manually via an in-house test console application (see section 6.5 Solver Launcher Console).

6.1 Generated Random Input Data

Three sets were generated with the following common features (detailed in section 6.4.3 Large-scale random data generation):

- Three platforms (satellites), each with the same number of orbits and the same revolution period;
- Orbital constraints (memory, energy, thermal) and global constraints (task budget, global budget) are restrictive enough so that they are easily exceeded;
- A mix of spot and large area targets, with many opportunities spread across many orbits;
- Individual opportunities can cover anything from almost nothing to the whole task area (a lot of overlaps). Their `probability` field is random but their `quality` field is fixed to 1; and
- Varying number of tasks between sets, with the number of orbits and total budget adjusted accordingly.

All three data sets are stored as JSON files. On top of that, a small Wang data file (not part of Wang data sets in the next section, although from the same source) was added for the sake of comparison. It differs slightly from the above data sets:

- Only spot targets tasks: each opportunity covers 100% of the task it is associated to;
- No budget restriction (task and global); and
- No thermal constraint.

The following table summarizes data sets:

Table 9: Generated Test Input Dataset

Name	Number of orbits (for each platform)	Number of tasks	Total number of opportunities	Total budget (k$)	Estimated Upper Bound	Data set file size (MB)
Small (Wang)	7	30	36	N/A	8.852	0.13
Medium	43	100	2080	750	67.52	8.35
Medium-large	100	300	16187	2300	210.1	180
Large	140	1000	72938	7500	700.6	2013

The upper bound is estimated according to formulas in section 2.1.2 Formulas.

Those data sets are available in Annex A of this document:

- *2066C.022-REP-01-OLCT Rev. 02 - Annex A - Generated test data.7z*

6.2 Wang data sets

Those test data sets come from the author of [1]. They share those common features:

- Three platforms with their own parameters (energy and slewing rates for instance);
- 120, 160 or 200 tasks with a few opportunities each, over 7 or 14 orbits per platform (21 or 42 total distinct orbits):
    - All combinations of the above, with 10 data set instances per combination.
- Only spot targets tasks: each opportunity covers 100% of the task it is associated to;
- Opportunities have varying probabilities but fixed quality (1.0);
- No budget restriction (task and global); and
- No thermal constraint.

For the purpose of comparing solvers, energy and memory constraints from Wang data sets are multiplied by 7.0 at run-time.

Those data sets are available in Annex B of this document:

- *2066C.022-REP-01-OLCT Rev. 02 - Annex B - Wang test data.7z*

6.3 Test bench

Solvers are launched manually via an in-house test console application (see section 6.5 Solver Launcher Console).

Functional correctness aside, solvers are expected to perform in a reasonable amount of time. What is reasonable or not is not specified in current uses cases and requirements. Also, running times depend on input data, and actual simulated data may not be a representative sample of future real-life data. Nevertheless, solvers are benchmarked to provide ballpark figures on their performance-wise usability. For the comparison, they run on a virtual machine featuring workstation performances:

- 8 virtual CPUs at 2.30GHz (from an Intel Xeon E5-2620 v3);
- 64 GB RAM; and
- Typical SSD on the hypervisor.

Also, JVM parameters are left at their default, except for:

- `-Xmx48G`: allow up to 48 GB for the memory heap for most demanding cases (never actually occurred).

Loading times for generated data set files (see section 6.4) are not included in each solver performance benchmark. For the record, they are roughly linear with file size:

(1) Wang data, Small and Medium data set: less than 1 second;

(2) Medium-large data set: 3 seconds; and

(3) Large data set: 25 seconds.

6.4 Data generator application

This application is not a deliverable and is only documented here for the purpose of future characterization of this data, compared to real-world data.

6.4.1 Application overview

This application can generate data to cover a few scenarios:

- Small (human-computable scale) and non-random data to test solvers functionalities (namely unit testing):
    - Generates an in-memory `DataSet` (see section 1.1.3 Data set) and serializes it into a JSON file that can be loaded by the solver launcher console (see section 6.5 Solver Launcher Console); and
    - See section 6.4.2 below.
- Somewhat random data with large numbers of tasks and opportunities to test solver performance and feasibility:
    - Generates an in-memory `DataSet` and serializes it into a JSON file that can be loaded by the solver launcher console (see section 6.5 Solver Launcher Console); and
    - See section 6.4.3 below.

- *Wang-format* random data: a small to large-scale data set, which follows the model described in [1]. Typically used to compare the Quest solver (see section 3 CPLEX-based Models/Solver) results with original results from [1]:
    - Generates a file in Wang format.

Source code is found in `DiscoverSolver` module, under `ca.gc.rddc.discover.solvers.utils.dataSet` package. However, given its in-house test/development tool status:

- Development time/effort was limited to:
    - No unit test;
    - In-code documentation limited to on-the-spot explanations and comments; and
    - Design/overview documentation limited to this section of this document.
- Low user configurability:
    - *Wang-format* data generator takes parameters from the command line; and
    - Other generators take parameters from static values in dedicated Java classes. Data is easily modifiable, yet it can only be changed in a development environment where the application can be rebuilt.

6.4.2 Small-scale Non-random Data Generation

This small-scale data set contains three similar tasks, each with four opportunities scattered on two orbits and two platforms. Unit tests typically take it as is or modify it slightly on the fly to test specific features.

Here are then main parameters and how they were chosen. See the source code for more details and remaining parameters:

Time frame:

- Two 100-minute orbits.

Satellites (platforms and sensors):

- Two platforms, with one sensor each; and
- Set-up time: 0.0 on the first, 35.0 on the second (set with *duplicated opportunity time offset*, see below).

Tasks:

- Same task duplicated two more times (three tasks on total), with respective priorities 0.5, 1.0 and 0.8.;
- Simple AOI to cover with one or more opportunities;
- Budget: $5,000 per task;
- Overall budget: $13,000; and
- Minimum relative coverage constraint: 0.65.

Opportunities:

- For each task, four opportunities covering partially the AOI:
    - Named a, b, c and d thereafter:

	Orbit	Platform	Start time	End time	Relative coverage	Cost
a	0	1	1000	1030	0.22	1800
b	0	0	2000	2060	0.42	1400
c	1	1	10000	10020	0.24	1000
d	1	0	11000	11010	0.70	2500

- From a task to the other generated opportunities are duplicated and then offset by 50.0 seconds:
    - A simulated transition time of 30.0 seconds is in effect only from opportunities of task 0 to opportunities of task 1. Other transitions have their transition time equal to 0.0:
        - Rationale: this makes some but not all transitions infeasible (only those where transition + set-up time > 50.0).
    - Start and end times were chosen so that the four opportunity graphs (see below) are different, even when merely duplicating and offsetting opportunities from a task to the other.
- Overlaps:
    - Opportunity coverages were “drawn” on a 50-unit grid, representing the AOI;
    - Simply counted relevant coverage/overlaps units, each worth 0.02 of relative AOI coverage:

.b.d	.b.d	.b.d	.b.d	.b.d	.b..	.b..	
.b.d	.b.d	.b.d	.b.d	.b.d	.b..	.b..	
.b.d	.b.d	.b.d	.b.d	.b.d	ab..	ab..	
...d	...d	...d	...d	...d	a...	a...	
..cd	..cd	..cd	..cd	...d	a...	a...	
..cd	..cd	..cd	..cd	...d	a...	a...	
..cd	..cd	..cd	..cd	...d	a...	a...	a...

For a minimal solver testing challenge, the above contains:

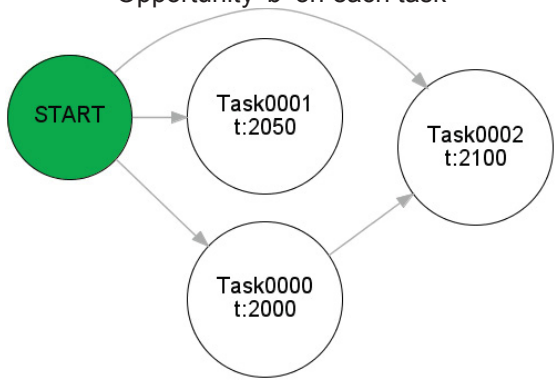
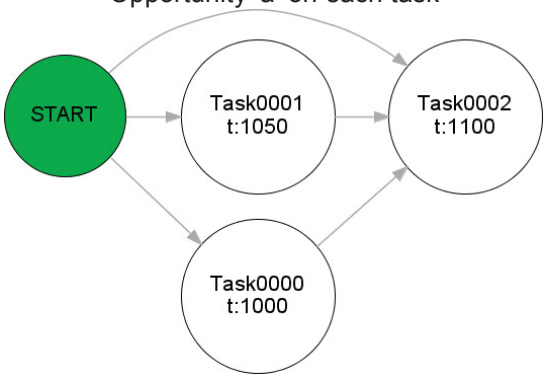
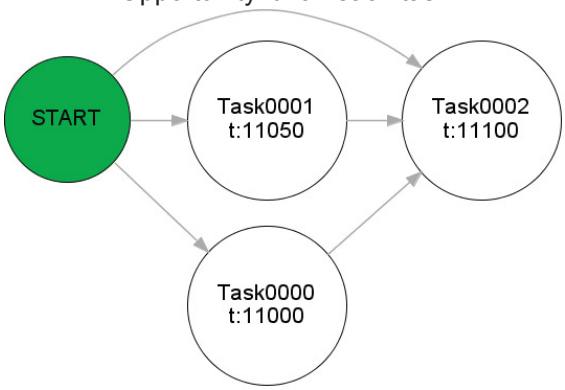
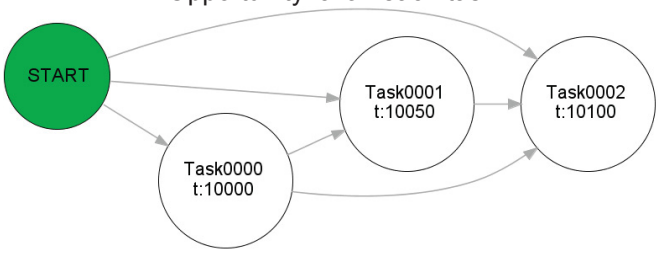
- Opportunity **d** fully overlaps opportunity **c**;
- No opportunity covers everything by itself:
    - Only a few opportunities combinations can meet the minimum task relative coverage requirement.

- Costs/budgets were set so that only two or three opportunities per task could be selected:
    - The best selection given cost constraint (opportunities a and d), is not the earliest opportunity; and
    - The best selection can't be done on all tasks given the overall budget.

This results in the following opportunity graphs:

- Note: actual generated opportunities all have unique IDs. They are named here a, b, c and d for each task for explanation purposes only.

Table 10: Examples of Opportunity Graphs

	Platform 0	Platform 1
Orbit 0	Opportunity 'b' on each task 	Opportunity 'a' on each task 
Orbit 1	Opportunity 'd' on each task 	Opportunity 'c' on each task 

6.4.3 Large-scale random data generation

Random data is generated with some constant parameters and several others controlling the amount of randomness. If follows a few key points:

- Starts with a few constants:
    - Time horizon and orbit period (for now, the same for all platforms);

- Number of tasks; and
- Platform-related: how much of them, their sensors, and nominal values (boundaries or averages) for some features. Those values are to be refined by random later (in per-orbit platform capacities for instance).

- Platforms and sensors are generated first;
- Tasks are then generated:
  - For now, tasks are "simple" tasks. They do not require to be composed from more than one subtask to be worth something (such as stereo imaging that has no value if only "side" of the task could be performed); and
  - About half (in current tests) of tasks are spot targets (every opportunity covers 100% of the task AOI), while the rest have large polygon AOI.
- On each orbit across the time horizon, platform capacities are computed with some randomness from their nominal feature values;
- On each orbit across the time horizon, and for each task, an opportunity can be added or not following some probability:
  - Some tasks have higher probability than others;
  - Start/end times are random but are quantized (rounded off) to increase likelihood of a time-based collision in solvers. Some jitter is then added to also allow for some "very close yet possible to do both" opportunities; and
  - Platform/sensors are assigned at random, although some opportunities are sometimes duplicated on the same platform to ***simulate*** the fact that CSIAPS or other system could return several opportunities for the same platform/task match. For instance, the imaging of the same target at a given time could be done with various sensor modes on the same platform.
- Opportunity coverage is random, but is also translated into a simulated coverage in a sort of bitmap (actually a plain boolean array):
  - For example, a 50% coverage would turn on half of the bitmap units;
  - All relative coverages use the same bitmap resolution (e.g. 100 units); and
  - Pair-wise opportunity coverage area overlap is then directly counted from a logical AND over two bitmaps (counting only corresponding units that are "on" on both bitmaps). This effectively simulates real area intersections.
- The rest of data is generated with controlled randomness.

6.4.4 Wang-format data generator

Here are the few steps used to generate data:

1. Determine |R| set of tasks; $V_{r0} = \text{Ran}_r$
2. Determine number of effective orbits $\rho_{\text{eff}} \rightarrow |R|/ \rho_{\text{eff}} = $ <#task/orbit>

3. Determine number of opportunities per tasks: k|SAT| (k|SAT| ≤ ρ_{eff})

   - k $\propto$ k(target task latitude | spatial distribution) = $\left(\frac{k_{\max}-1}{\pi/2}\right) latitude + 1$

4. $\{O_\rho\}$ construction: opportunity distribution per orbit ρ $(1..\ \rho_{eff})$:

   For $r \in R$
   j=0
   For i = 1.. k|SAT|
   $p\ (o_i(r) \in \rho \in (1..\ \rho_{eff}) / \{\rho\_o_1, \rho\_o_2, \ldots, \rho\_o_{i-1}\}) = 1/(\rho_{eff} - j)$ –populate ρ
   j=j+1
   end for i
   end for *r*
   k|SAT| /ρ_{eff} = # opp task *r*/orbit
   |R| (k|SAT|) /ρ_{eff} = # opp/orbit = | O_ρ |≤ $T_\rho / \left|\bar{d}_\rho\right|$

5. Contention '*c*'/complexity/connectivity proportional to $\alpha \rightarrow 1$ large, ρ_{eff} small:

$$Given: R, \rho_{eff}, k|SAT|; \quad O_\rho, contention\ c(\alpha), T_{\max}$$

$$\alpha, T_{\max} \rightarrow \bar{d}, T_{\max} \geq d_{\max}, d_{\min} = 2\alpha T_{\max} - d_{\max}$$

$$\alpha T_{\max} \leq d_{\max} \leq \min(2\alpha T_{\max}, T_{\max})\ as\ d_{\max} \geq \bar{d}_\rho,\ d_{\min} \geq 0$$

$$\Rightarrow d_{\min} = \max(0, 2\alpha T_{\max} - d_{\max})$$

$$d_{\max} = [\min(2\alpha T_{\max}, T_{\max}) - \alpha T_{\max}]ran + \alpha T_{\max}$$

$$\frac{\overbrace{|O_\rho|}^{1/\alpha}}{c} \bar{d}_\rho = \overbrace{T_{\max}}^{\sim 20 \min} \leq T_\rho \Rightarrow \bar{d}_\rho \leq T_\rho / |O_\rho| \quad \frac{1}{\alpha}\ schedulable\ opp\ by\ orbit$$

$$\frac{\bar{d}_\rho}{\alpha} = T_{\max}$$

$$d_o = (d_{\max} - d_{\min})Ran + d_{\min} = 2(\bar{d}_\rho - d_{\min})Ran + d_{\min} \qquad o \in O_\rho$$

$$\bar{d}_\rho = \frac{1}{2}(d_{\max} + d_{\min}) = \alpha T_{\max}$$

$$\theta_o = 2\theta_{\max\,\rho}\ Ran - \theta_{\max\,\rho} \quad o \in O_\rho$$

$$ts_o = \left(T_{\max} - d_{\max}\right)Ran + \left(\overbrace{t_\rho}^{orbit\,\rho=cycle\,t} - 1\right)T_\rho, \quad o \in O_\rho, \quad \rho(sat) = (t-1)\left|SAT\right| + sat$$

$$\left(ts_o \in \left[\left(t_\rho - 1\right)T_\rho, \left(t_\rho - 1\right)T_\rho + T_{\max} - d_{\max}\right]\right)$$

$$\Rightarrow T_{\max} - d_{\max} \geq 0 \Rightarrow \frac{\bar{d}}{\alpha} \geq d_{\max} \Rightarrow \left(\alpha = 1 \rightarrow \bar{d} = d_{\max} = T_{\max} = cons\tan t\right)$$

6.5 Solver Launcher Console

This is not a deliverable and is only minimally documented here.

6.5.1 Overview

The console is a small command-line Java application. Its sole purpose is to call one of the implemented solvers with a given data set and some options. It is a temporary solution, expected to be replaced with an API/service interacting with CSIAPS.

In its current state:

- Application is launched from a development environment:
  - It is not deployable;
  - Runs the class `ca.gc.rddc.discover.solvers.utils.console.App` as a Java application; and
  - Runs it without arguments to get help on valid command-line parameters.
- Application reads data set from either one of:
  - Native JSON serialized data set (namely, from generated data set, see section 6.4 Data generator application);
  - Foreign text data samples (conforming to model in [1]) converted on-the-fly; or
  - Wang-style data generated on the fly.
- One of the available solvers is instantiated, according to the command-line options;
- Data sets and additional command-line options are forwarded to the solver;
- Solver is run and generates a “collection plan”;
- A rough report of that collection plan is printed to the console, including:
  - All selected opportunities, sorted by platform then by orbit;

- All selected opportunities (same as above), sorted by tasks along with relative coverage and cost for that task;
- A summary of the collection plan (*objective value* over its estimated upper bound); and
- For example:

```
Plan, by platform and orbit:
P0, orbit 0: 1:{Opp00001:2000, }
P0, orbit 1: 2:{Opp00007:11050, Opp00011:11100, }
P1, orbit 0: 2:{Opp00004:1050, Opp00008:1100, }
P1, orbit 1: 1:{Opp00002:10000, }

Plan, by task:
Task ID: priority, relative coverage, cost, selectedOpportunities
Task0000: 0.500, 0.660,   2400.00, 2:{ Opp00001:0, Opp00002:1, }
Task0001: 1.000, 0.920,   4300.00, 2:{ Opp00007:1, Opp00004:0, }
Task0002: 0.800, 0.920,   4300.00, 2:{ Opp00011:1, Opp00008:0, }

Summary
Plan vlaue: 1.986 / 2.300 (86.35%)
Cost:    11000.00
```

6.5.2 Design and Implementation

Code is split into a few self-explanatory classes in `DiscoverSolver` module, all under a few `ca.gc.rddc.discover.solvers.utils.console.*` test packages (under `/src/test/java`). Those mostly glue application command-line options to solver classes. See source code and its inline documentation for more details.

7 SOLVERS COMPARATIVE RESULTS

MY-PICC, GATHER and QUEST solvers were compared using Wang data sets (see section 6.2 Wang data sets) since they all could tackle this kind of data. MY-PICC and GATHER can address a broader set of problems (namely large area targets and their individual opportunities, as well as more constraints), and so can QUEST (mostly its ability to take on input pre-defined task plans made from several opportunities across the data set). Wang data sets are therefore the common denominator.

Tests were performed on the test bench described in 6.3 Test bench.

7.1 DISCOVER vs WANG Objective Function

As explained in section 2.1 Mathematical model, the DISCOVER objective function covers tasks value as well as opportunities probability, quality and relative coverage of the task area. It fails, however, to compute a proper value whenever there are more than two opportunities for the same task. It rather computes an approximation which is equal or lower than the theoretical value. On the other hand, the WANG objective function, a variant of the former, considers only task value and opportunities probability (which is enough for Wang data sets) but does compute the proper collection value for such data, no matter how many opportunities are selected for a task.

QUEST translates the DISCOVER objective function (see section 2.1.2) into a CPLEX model. As such, most of the source code itself of the solver implements this translation. Other solvers simply use a direct implementation (see section 2.2) of these objective function formulas and can thus accept any objective function that is decomposed into the same sub-objective function, such as the WANG objective function variant.

Solver comparison was therefore performed using both objective functions, albeit with the following caveats:

- DISCOVER objective function:
    - All solvers try in their own way to maximize the collection value (or parts of it), although attempts become suboptimal whenever there are three or more opportunities on any given task; and
    - Collection value of the result exhibits the same sub-optimality.
- WANG objective function:
    - QUEST solver tries to maximize value using DISCOVER objective function because it is hard-coded to do so. Final collection value is, however, computed using WANG objective function:
        - Resulting collection value is equal or greater than it would be with DISCOVER, but equal or less than it would be, had a WANG objective function variant of QUEST been available.
    - Other solvers use the WANG objective function both for maximizing the collection value and computing it at the end. Result is thus optimal in regard to each solver, but comparison with QUEST is not perfect.

7.2 Solvers configurations

The following configurations were compared:

- QUEST:
    - The latest available version.

- MY-PICC:
    - Both variants are tested: Time-Weighted (TW) and Plain-Value (PV).
- GATHER:
    - Tested with two configurations: default and *slow* variant (as described in sections 5.1.9.1 and 5.1.9.2 respectively); and
    - Given the random nature of this solver, run times and collection values are averaged over 5 runs.

7.3 Result Summary

Full results are available in Annex C of this document:

- *2066C.022-REP-01-OLCT Rev. 02 - Annex C – Solver comparison n Wang data sets.xlsx*

Note that numerical collection values in this annex were all multiplied by 10 at the end, so as to be comparable with [1]. Indeed, our solvers normalize the task nominal value (within [0.1, 1] in this case), while Wang data sets use [1, 10]. This obviously does not affect relative value comparison between solvers.

Table 11: Comparison summary using DISCOVER objective function

	QUEST	GATHER (default)	GATHER (slow)	MY-PICC (TW)	MY-PICC (PV)
Average Collection Value relative delta over QUEST		0.1%	3.2%	-19.6%	-35.5%
Average run time (second)	0.320	0.761	18.250	0.007	0.006

Table 12: Comparison summary using WANG objective function

	QUEST	GATHER (default)	GATHER (slow)	MY-PICC (TW)	MY-PICC (PV)
Average Collection Value relative delta over QUEST		-2.1%	0.8%	-21.2%	-37.3%
Average run time (second)	0.309	0.577	13.352	0.006	0.006

7.4 Discussion

MY-PICC (both variants) is incredibly faster than others, but fails to give good results which such data sets, worse that it was with random generated data sets. It might have to do with inherent characteristics of the set, such as opportunities temporal density, number of opportunities per task, the lack of discriminating values (only task value and opportunity probability), etc. This could be investigated at a later time.

GATHER *default* performs well with reasonable running times (about twice as long as QUEST), but the *slow* variant is much slower (20-25 times) for a marginally better collection value (about 3%). Depending on actual requirements in the fields, both are nevertheless viable options.

Surprisingly, GATHER performs better than QUEST on some data sets, even on average for the DISCOVER objective function. It was expected that QUEST could find the single optimal collection value, but obviously it doesn't. This may be caused by a trade-off between result optimality and resource usage (execution time, memory). Also given the suboptimal QUEST result when using WANG objective function, it was expected that other solvers would fare slightly better (compared to their respective difference with QUEST using DISCOVER objective function), but it was the opposite by about 2%. This warrants future investigation.

8 BIBLIOGRAPHY

[1] J. Wang, E. Demeulemeester, D. Qiu and D. Liu, "Exact and inexact scheduling algorithms for multiple earth observation satellites under uncertainties of clouds," *European Journal of Operational Research,* 2015.

[2] A. Globus, J. Crawford, J. Lohn and A. Pryor, "A comparison of techniques for scheduling fleets of earth-observing," *J Oper Res Soc,* vol. 56, no. 8, pp. 962-968, 2003.

[3] F. Nelson, "Scheduling optimization for imagery satellite constellations using column generation," PhD Thesis, George Mason University, Department of Systems Engineering and Operations Research, Volgenau School of Engineering, 2012.

[4] N.-G. Hall, "Maximizing the value of a space mission," *Magazine MJ, Eur J Oper Res,* vol. 78, no. 2, pp. 224-241, 1994.

[5] V. Gabrel and D. Vanderpooten, "Enumeration and interactive selection of efficient paths in a multiple criteria graph for scheduling an earth observing satellite," *Eur J Oper Res,* vol. 139, no. 3, pp. 533-542, 2002.

[6] E. Bensana, G. Verfailliek, J.-C. Agnese, N. Bataille and D. Blumestein, "Exact and inexact methods for the daily management of an earth observation satellite," *Proceeding of the international symposium on space mission operations and ground data systems,* vol. 4, pp. 507-514, 1996.

[7] V. Gabrel, A. Moulet, C. Murat and V.-T. Paschos, "A new single model and derived algorithms for the satellite shot planning problem using graph theory concepts," *Ann Oper Res,* vol. 69, pp. 115-134, 1997.

[8] J. Wolfe and S.-E. Stephen, "Three scheduling algorithms applied to the earth observing systems domain," *Manag Sci,* vol. 46, no. 1, pp. 148-168, 2000.

[9] M. Vasquez and J. Hao, "A logic-constrained knapsack formulation and a tabu algorithm for the daily photograph scheduling of an earth observation satellite," *Comput Optim Appl,* vol. 20, no. 2, pp. 137-157, 2001.

[10] M. Lemaître, G. Verfaillie, F. Jouhaud, J.-M. Lachiver and N. Bataille, "Selecting and scheduling observations of agile satellites," *Aerospace Science and Technology, 6,* p. 367–381, 2002.

[11] A. Globus, J. Crawford, J. Lohn and A. Pryor, "A Comparison of Techniques for Scheduling Earth Observing Satellites," in *IAAI Emerging Applications*, 2004.

[12] W. Lin, D. Liao, C. Liu and Y. Lee, "Daily Imaging Scheduling of an Earth Observation Satellite," *IEEE Trans Syst Man Cybern A,* vol. 35, no. 2, pp. 213-223, 2005.

[13] W. Lin and S. Chang, "Hybrid algorithms for satellite imaging scheduling," *Proceeding of the IEEE international conference on systems, man and cybernetics,* vol. 3, pp. 2518-2523, 2005.

[14] N. Bianchessi, G. Laporte et al, "A heuristic for the multi-satellite, multi-orbit and multi-user management of earth observation satellites," *Eur. J. Oper. Res,* vol. 177, no. 2, p. 750–762, 2007.

[15] H. Wang, M. Xu, R. Wang and Y. Li, "Scheduling earth observing satellites with hybrid ant colony optimization algorithm," in *Proceeding of the international conference on artificial intelligence and computational intelligence*, AICI'09, 2009.

[16] G. Wu, J. Liu, M. Ma and D. Qiu, "A two-phase scheduling method with the consideration of task clustering for earth observing satellites," *Computers and Operations Research,* vol. 40, no. 7, pp. 1884-1894, 2013.

[17] D. Liao and Y. Tang, "Imaging order scheduling of an earth observation satellite," *IEEE Trans Syst Man Cybern,* vol. 37, no. 5, pp. 794-802, 2007.

[18] G. Verfaillie, M. Lemaitre and T. Schiex, "Russian doll search for solving constraint optimization problems," in *Proceeding of the thirteenth national conference on Artificial intelligence* , AAAI'96, pp.181-187, 1996.

[19] M. Lemaitre, G. Verfaillie, F. Jouhaud, J.-M. Lachiver and N. Bataille, "How to manage the new generation of agile earth observation satellites," in *Proceeding of the international symposium on artificial intelligence, robotics and automation in space*, ISAIRAS'00, 2000.

[20] V. Gabrel and C. Murat, "Mathematical programming for earth observation satellite mission planning," in *T.-A. Ciriani, G. Fasano, S. Gliozzi, R. Tadei, Operations research in space and air*, Springer, 2003, pp. 103-122.

[21] V. Gabrel, "Strengthened 0-1 linear formulation for the daily satellite mission planning," *J Comb Optim,* vol.

11, no. 3, pp. 341-346, 2006.

[22] M. Vasquez and J. Hao, "Upper bounds for the SPOT 5 daily photograph scheduling problem," *J Comb Optim,* vol. 7, no. 1, pp. 87-103, 2003.

[23] T. Benoist and B. Rottembourg, "Upper bounds for revenue maximization in a satellite scheduling problem," *4OR-Q J Oper Res,* vol. 2, no. 3, pp. 235-249, 2004.

[24] J. Berger, "Implement Plan Sched 2016-2017 v2 16 Dec 16," RDDC, Valcartier, 2017.

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 (Québec) G1P 4P5
Canada

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—Open-Loop Collection Tasking

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

Florea, M.; Giasson, E.

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.)
September 2018	96	24

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-C168	

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