On-board decision-making on data downloads

G. Verfaillie, G. Infantes, and **M. Lemaître**
ONERA, Toulouse, France
FirstName.LastName@onera.fr

N. Théret and **T. Natolot**
CNES, Toulouse, France
FirstName.LastName@cnes.fr

Abstract

In this paper, we present a new planning problem which arises when data download windows and rate are limited and when the volume of data actually generated by instrument activation on board a spacecraft is uncertain and only known after acquisition and data analysis, compression, and memorization. We present several pragmatic approaches to this problem and results of experiments on a realistic scenario.

Introduction

Many space missions are observation missions which consist in using space observation instruments to collect data, to record it, and to download it to ground mission centers and to final users who use it to extract relevant information.

In many cases (Earth orbiting satellites, spacecraft orbiting other planets, or rovers at the surface of these planets), the temporal windows that are available for data downloading are limited. Moreover, the data download rate may be strongly limited. This is especially true for deep space exploration missions.

In many cases too, the volume of data to be downloaded is not precisely known in advance. Such a situation arises with classical Earth observation satellites due to a variable compression rate of images. It arises with advanced more autonomous Earth observation satellites that are equipped with on-board data analysis software (see for example the experimental EO-1 mission (Chien et al. 2004)) able for example to discard images that are useless because of the presence of too many clouds. It arises too with future Mars exploration rovers that are designed to be able to perform opportunistic scientific observations (see for example (Castano et al. 2007; Thompson, Smith, and Wettergreen 2008; Woods et al. 2009)). Finally, it arises with Earth surveillance missions, such as for example electromagnetic intelligence missions, where the result of surveillance is not known in advance.

In all these cases, it becomes difficult to build data download plans off-line on the ground, as it is usual for many space missions. If maximum volumes of data are taken into account to build such plans, download windows may be under-used and mission return may be needlessly limited. If expected volumes are taken into account, some data may be lost due to actual volumes greater than expected ones. A solution consists in building data download plans on-line on board when actual volumes of data are known.

In this paper, we report the main results of a study that has been carried out by CNES (French Space Agency) and ONERA (French Aerospace Lab) in the context of an Earth surveillance mission. In this context, it is assumed that the surveillance activities are planned off-line before execution in mission centers on the ground. However, the volume of data generated by these activities is not known in advance. Only, a probability distribution on the actual volume is available for each surveillance activity. This distribution may have a large variance (typically, the volume of data generated by a surveillance activity may go from 1 to 1000). The actual volume of data generated by a surveillance activity is only known at the end of this activity, when all the data is recorded in the satellite mass memory. Data downloading is only possible within visibility windows of ground reception stations.

After problem formalization and analysis, we describe several candidate approaches, including planning on the ground, decision rules to be applied on board, planning on board, and sampling-based decision-making on board. All these approaches have been implemented and compared on a real-size realistic scenario. We present and discuss experimental results.

Problem data

The problem data associated with a ground planning horizon H (typically one day) is the following one:

1. a set of Ac surveillance activities over H, planned off-line on the ground, numbered from 1 to Ac, and ordered according to an increasing ending time;
2. a set of Pr priority levels, numbered from 1 to Pr, from the highest to the lowest;
3. a set of St ground reception stations, numbered from 1 to St; the first one is the main station associated with the mission center; the others are secondary and data transmitted to one of them must be then transmitted to the mission center using a ground communication network;
4. a set of Wi visibility windows over H, numbered from 1 to Wi, ordered according to an increasing starting time;

5. a data downloading rate Dr;
6. for each surveillance activity $a \in [1..Ac]$:
   (a) a starting time Sa_a;
   (b) an ending time Ea_a;
   (c) a probability distribution Pv_a on the volume generated by a; probability distributions associated with different activities are assumed to be independent from each other;
   (d) a maximum volume Vm_a and a maximum download duration $Dm_a = Vm_a/Dr$; memorization is stopped when this volume is reached;
   (e) an expected volume Ve_a and an expected download duration $De_a = Ve_a/Dr$;
   (f) an actual volume Va_a and an actual download duration $Da_a = Va_a/Dr$, only known after data memorization;
   (g) a deadline Dd_a; beyond this deadline, downloading a has no utility because data is too old;
   (h) a priority $Pr_a \in [1..Pr]$; activities of priority p have absolute priority over any set of activities of priority $p' > p$;
   (i) a weight We_a; weights are used to express relative preferences at the same priority level and are assumed to be additive;
7. for each station $s \in [1..St]$, a duration Dt_s of data transfer from s to the mission center, with $Dt_1 = 0$.
8. for each window $w \in [1..Wi]$:
   (a) an associated station $St_w \in [1..St]$;
   (b) a starting time Sw_w;
   (c) an ending time Ew_w;
9. for each pair of windows $w, w' \in [1..Wi]$, a boolean $Ov_{w,w'}$ equal to 1 if and only if both windows overlap with, for all $w \in [1..Wi]$, $Ov_{w,w} = 1$;
10. a half-life period Hl used to express how download utility decreases exponentially with download time; if dl_a is the duration from the time at which activity a ended and the time at which data associated with activity a is delivered to the mission center, we define the actual utility of a as $We_a \cdot 2^{-dl_a/Hl}$: equal to We_a if data would be immediately delivered, divided by 2 if data is delivered Hl after the end of a, by 4 if it is delivered $2Hl$ after the end of a, by 8 if it is delivered $3Hl$ after the end of a, and so on.

Data download problem

Decision variables The problem is to decide, for each acquisition $a \in [1..Ac]$, on two variables:

1. an integer variable $w_a \in [0..Wi]$ which represents the window within which a is downloaded, equal to the default value 0 when a is not downloaded over H;
2. a real variable d_a which represents the starting time of the download of a, when $w_a \neq 0$.

Constraints These variables must satisfy the following constraints:

Data associated with an activity a cannot be downloaded before the end of a:

$$\forall a \in [1..Ac], \quad (w_a \neq 0) \rightarrow (d_a \geq Ea_a) \tag{1}$$

Data associated with an activity a must be delivered before its deadline :

$$\forall a \in [1..Ac], \quad (w_a \neq 0) \rightarrow (d_a + Da_a + Dt_{St_{w_a}} \leq Dd_a) \tag{2}$$

Data must be downloaded within the chosen window :

$$\forall a \in [1..Ac], \quad (w_a \neq 0) \rightarrow (Sw_{w_a} \leq d_a \leq Ew_{w_a} - Da_a) \tag{3}$$

Downloads cannot overlap :

$$\forall a, a' \in [1..Ac] \mid a < a', \quad ((w_a \neq 0) \wedge (w_{a'} \neq 0) \wedge (Ov_{w_a,w_{a'}} = 1) \rightarrow ((d_a + Da_a \leq d_{a'}) \vee (d_{a'} + Da_{a'} \leq d_a)) \tag{4}$$

Optimization criterion The optimization criterion is a vector of utilities, with one utility for each priority level. To compare two solutions, it suffices to compare the two associated vectors in a lexicographic way, from the highest utility 1 to the lowest Pr.

For each priority level $p \in [1..\Pr]$, the associated utility U_p is defined as follows:

$$U_p \stackrel{\text{def}}{=} \sum_{a \in [1..Ac] \mid (\Pr_a = p) \wedge (w_a \neq 0)} u_a \tag{5}$$

For each activity $a \in [1..Ac]$ that is downloaded ($w_a \neq 0$), the associated utility u_a is defined as follows:

$$u_a \stackrel{\text{def}}{=} We_a \cdot 2^{-dl_a/Hl} \tag{6}$$

where dl_a is the difference between the time at which data associated with activity a is delivered to the mission center and the time at which activity a ended: $dl_a \stackrel{\text{def}}{=} d_a + Da_a + Dt_{St_{w_a}} - Ea_a$.

Impact of uncertainty The difficulty is that decisions must be made upon downloads without knowledge of the actual volumes of the data that is not memorized yet. Concerning the constraints, they must be satisfied whatever data volumes are. It is always possible because any data may be not downloaded. Concerning the criterion, the objective is to maximize its expected value according to the probability distributions over data volumes.

Problem analysis

Generally speaking, the data download problem without uncertainty can be seen as a (multi) knapsack problem (Kellerer, Pferschy, and Pisinger 2004), where visibility windows are sacks, downloads are objects to be placed into

sacks, and download durations are object weights. It however differs from the (multi) knapsack problem if we consider the temporal constraints (earliest and latest download times, download utility function of download time, possible overlaps between windows). In spite of these differences, it may be relevant to use efficient knapsack heuristics to solve the data download problem. One of these heuristics consists in sorting objects according a decreasing utility density (ratio between utility and weight) and in inserting them into sacks according this order, until no object can be added. Such a heuristics will be widely used in the algorithms that are described in the next section.

When the volumes of data and thus the download durations are uncertain, the data download problem becomes close to the stochastic (multi) knapsack problem. In fact, there are several versions of the stochastic knapsack problem, according to what is uncertain in the problem definition (maximum weight of the sack, weight of the objects, gains associated with the objects) and the time at which uncertainty disappears. The closest version to our problem is described in (Dean, Goemans, and Vondrak 2004): uncertain object weights, actual weights known when objects are placed into the sack. A small difference is that, in our problem, volumes are not known at the download time, but before, at the memorization time. Unfortunately, (Dean, Goemans, and Vondrak 2004) and other studies of the stochastic knapsack problem do not propose specific optimization algorithms. They simply analyze either theoretically or empirically the quality of the solutions produced using stochastic variants of classical heuristics, such as the one presented above (obtained by replacing object weights by expected ones).

More generally, the data download problem with uncertain volumes can be modeled as an MDP (Markov Decision Problem (Puterman 1994)) whose main features are the following ones:

- decision steps are starts of visibility windows, ends of activities, or ends of downloads;
- the system state is defined by the current time t and, for each activity $a \in [1..Ac]$, its current volume v_a in the mass memory, which is null before the end of a and after download, and positive otherwise;
- the chosen action is defined by an activity $ac \in [0..Ac]$ which represent the data to be downloaded and a window $wc \in [0..Wi]$ within which data will be downloaded; values 0 are used to represent the absence of download in order to wait for the beginning of a visibility window or the end of an activity;
- the system state resulting from an action ac is uncertain because of the uncertain volumes of data memorized between the current time and the next one, which is the end of the chosen download or of the waiting period (uncertainty does not come from actions, but from the environment).

The result is an MDP with a variable, but bounded, number of decision steps, with continuous states, and with discrete actions. To compute off-line on the ground an optimal or near-optimal policy (something which would say which action to choose in any system state), approximate algorithms such as for example HRTDP (Hybrid Real-Time Dynamic Programming (Teichteil-Königsbuch and Infantes 2009)) could be used. HRTDP is a hybrid version of RTDP (Barto, Bradtke, and Singh 1995). RTDP samples the set of possible system trajectories to estimate the optimal value of each state and to identify the associated optimal action. HRTDP does the same by using regressors to represent optimal values of states and classifiers to represent associated optimal actions.

On top of the same MDP modeling, other approximate approaches, such as Hindsight Optimization (Chong, Givan, and Chang 2000) or Online Stochastic Optimization (Hentenryck and Bent 2006) could be used on-line on board to choose the best action to be performed in the current state. Both sample the set of possible system trajectories over a limited horizon ahead the current state.

Another way of choosing on-line the best action to be performed, inspired from the Model Predictive Control approach (Garcia, Prett, and Morari 1989) in Automatic Control, would consist in planning over a limited horizon ahead the current state, assuming that all the unknown volumes will be equal to their expected value, and in selecting the first action of the resulting plan.

Finally, another way of choosing still on-line the best action to be performed would consist in using well designed decision rules.

In the next section, we describe more precisely, the approaches we have implemented and experimented.

Several approaches

Planning on the ground

We first develop a reference approach which consists in building a download plan off-line on the ground without knowledge of the actual volumes. To be sure that this plan be executable, we consider a first variant, referred to as $PG1$, where all volumes are assumed to be maximum. To be sure that at least data associated with highest priority activities be downloaded, we consider a second variant, referred to as $PG2$, where volumes are assumed to be maximum when inserting activities of priority 1 and 2 (the most important) and to be equal to their expected value when inserting activities of priority 3 (the less important). With the latter variant, downloads of data of priority 3 may be lost in case of too large actual volumes.

To build such a plan, we use a non-chronological greedy algorithm based on classical knapsack heuristics. The algorithm starts from an empty plan. At each step, it chooses an activity a (i) whose data can be downloaded using one of the remaining windows, (ii) that is of highest priority, and (iii) that, at equal priority, maximizes the utility density (ratio between (1) a's utility if data is delivered at the earliest time and (2) the duration of a's data download). Once a is chosen, data download is inserted in the current plan in such a way that data be delivered to the mission center at the earliest time and the remaining windows are accordingly reduced (sometimes split into sub-windows). The algorithm

stops when all activities have been considered or no more download can be inserted in the current plan.

Applying decision rules on board

To make on-line decisions, the most popular approach consists in designing expert decision rules. We designed two rules to be applied on-line on board. The second one is more sophisticated, but more time consuming than the first one.

First decision rule The first decision rule, referred to as $DR1$, consists in choosing at each decision step an activity a (i) whose data can be immediately downloaded using one of the current windows (activity already ended), (ii) that is of highest priority, and (iii) that, at equal priority, maximizes the utility density (see above). Once a is chosen, the chosen window is, among the current ones, the one that allows data to be delivered at the earliest time.

Second decision rule The second decision rule, referred to as $DR2$, consists in choosing at each decision step an activity a (i) whose data can be downloaded using one of the current windows (activity possibly not ended yet), (ii) that is of highest priority, and (iii) that, at equal priority, maximizes the utility density (ratio between (1) a's utility if data is delivered at the earliest time and (2) the duration from the current time to the end of a's data download *i.e.*, the time spent if a is chosen). Once a is chosen, the chosen window is, among the current ones, the one that allows data to be delivered at the earliest time.

This rule allows non immediate downloads to be chosen and thus to wait for the end of an activity to download associated data. Moreover, when the chosen download is not immediate, the rule searches recursively for activities whose data could be downloaded between the current time and the beginning of a's data download, until no such activity exists. For activities already ended, the algorithm considers the actual data volumes. For the others, it considers the expected ones.

Planning on board

The main drawback of the first approach (planning on the ground) is that actual volumes are unknown at the planning time.

As done in Model Predictive Control (Garcia, Prett, and Morari 1989), it is however possible to plan (and replan) on-board each time over a limited horizon ahead. To build such plans, it is possible to use the same non-chronological greedy algorithm we use for planning on the ground and to call it with actual volumes for activities already ended and expected ones for the others.

We developed two versions of this approach. In the first one, referred to as $PB1$, a new plan is built each time an activity ends and information about its actual volume is available. In the second one, referred to as $PB2$, a new plan is built at each decision step

An on-board implementation of both versions could be reasonably considered, because the greedy planning algorithm is very efficient and called each time only over a limited horizon.

Sampling-based decision-making on board

On-line stochastic optimization The previous approach (planning on board) can be improved by producing at each decision step samples of the future over a limited horizon ahead (possible volumes for activities not ended yet, randomly generated according to the known probability distributions), by considering each of these samples as certain information, by planning on this basis, and by selecting the action (download or waiting action) that appears the maximum number of times as the first action in the produced plans.

We refer to this approach, inspired from Online Stochastic Optimization (Hentenryck and Bent 2006), as $SB1$. Clearly, this approach is more time consuming than the previous one, because several plans (one for each sample) must be built at each decision step.

Hindsight optimization The previous approach (on-line stochastic optimization) can be further refined by considering at each decision step all the possible actions (download or waiting action), by evaluating each of them on the basis of sampling and planning, and by selecting the most valuable action.

To evaluate an action a, we produce samples of the future over a limited horizon ahead the end of a (possible volumes for activities not ended yet), we consider each of these samples as certain information, we plan on this basis, and then we consider the mean utility of the produced plans (including a's utility; in fact the vector of the mean utilities, one for each priority level). An action with maximum mean utility is then selected.

We refer to this approach, inspired from Hindsight Optimization (Chong, Givan, and Chang 2000), as $SB2$. Clearly, this approach is still more time consuming because several plans (one for each sample) must be built for each possible action at each decision step.

Experimental results

The approaches described in the previous section have been implemented and experimented on a real-size realistic scenario built by CNES.

This scenario covers a one-week horizon. It involves 1484 surveillance activities: 140 of priority 1, 420 of priority 2, and 924 of priority 3 (the lowest). All activities have the same weight (1). Probability distributions on volumes generated by activities are assumed to be all Gaussian, with varying minimum, maximum, and mean values, and varying standard deviations. For each activity, its actual volume is fired according to the associated probability distribution.

Planning on board ($PB1$, $PB2$, $SB1$, and $SB2$) is systematically performed over a 30 minute horizon ahead. For $SB1$, 50 samples are considered at each decision step. For $SB2$, 10 samples are considered for each possible action at each decision step.

Table 1 shows, for each approach, the following results:

- the number of downloads performed over the one-week horizon;
- the utilization percentage of the download windows;

	downloads	*utilization*	U_1	U_2	U_3
$PG1$	560	31.32%	0.763	0.468	0.256
$PG2$	807	62.78%	0.764	0.468	0.500
$DR1$	850	77.81%	0.926	0.850	0.345
$DR2$	1104	95.64%	0.925	0.853	0.592
$PB1$	1121	95.77%	0.925	0.857	0.608
$PB2$	1110	95.86%	0.925	0.861	0.596
$SB1$	1104	96.75%	0.926	0.873	0.589
$SB2$	1153	97.55%	0.926	0.874	0.615

Table 1: Experimental results (number of downloads, utilization percentage of the download windows, and utility at the three priority levels): PG for *Planning on the Ground*, DR for Decision Rules on board, PB for *Planning on Board*, and SB for *Sampling-based decision-making on Board*

$DR1$	$DR2$	$PB1$	$PB2$	$SB1$	$SB2$
1	5.52	3.03	6.68	1967.16	4418.39

Table 2: Computing time relatively to that of $DR1$, for all approaches that require on-board computing

- the resulting utility at the three priority levels.

Table 2 shows, for each approach that requires on-board computing, the computing time relatively to that of the simplest approach ($DR1$, first decision rule on board). Making one decision with $DR1$ takes about 0.3 millisecond on a 3GHz processor.

$PG1$ (Planning on the Ground, using maximum volumes) produces the worst results: only 31% of the download windows are used. $PG2$ (Planning on the Ground, using maximum volumes only for activities of priority 1 and 2) produces better results for activities of priority 3 (the lowest): 63% of the download windows are used.

$DR1$ (first Decision Rule on board) produces better results for activities of priority 1 and 2 (the highest): 78% of the download windows are used. However, $DR2$ (second Decision Rule on board) performs better for activities of priority 3 (the lowest): 96% of the download windows are now used, at the cost of a computing time only multiplied by 5.

The results produced by $PB1$ and $PB2$ (Planning on Board) are a bit better, but not significantly better. Computing times remain of the same order of magnitude.

The results produced by $SB1$ and $SB2$ (Sampling-based decision-making on Board) are similar. Among all the experimented approaches, the best results, in terms of utilization percentage and of utility, are produced by $SB2$. However, these results are obtained at a very high cost in terms of computing time (some thousands times the computing time taken by $DR1$).

These results should be however consolidated by firing several scenarios in terms of actual volumes of data (only one scenario has been fired for the moment).

Conclusion

A first conclusion is that, even if the problem considered remains relatively simple (only data download planning), there is no unique technical solution. Several approaches may be considered. Each of them has its strengths and weaknesses in terms of implementation, computing time, and result quality.

A second conclusion is that approaches based on decision-making on the ground ($PB1$ and $PB2$) are clearly outperformed by approaches based on decision-making on board, when actual volumes of data are known. The former (decision-making on the ground) are outperformed in terms of result quality: number of downloads, utilization of the download windows, and utility.

Among the latter (decision-making on board), applying sophisticated decision rules ($DR2$) or planning on board ($PB1$ or $PB2$) seem to achieve the right trade-off between result quality and computing time. They have in common to look ahead one way or another before making immediate decisions, whereas the first decision rule ($DR2$) does not look ahead and produces results of significantly worse quality. However, it seems to be difficult to perform significantly better than $DR2$, $PB1$, or $PB2$, and more sophisticated sampling-based approaches ($SB1$ and $SB2$) produce slightly better results at a prohibitive cost in terms of computing time.

References

[Barto, Bradtke, and Singh 1995] Barto, A.; Bradtke, S.; and Singh, S. 1995. Learning to Act using Real-Time Dynamic Programming. *Artificial Intelligence* 72(1-2):81–138.

[Castano et al. 2007] Castano, R.; Estlin, T.; Anderson, R.; Gaines, D.; Castano, A.; Bornstein, B.; Chouinard, C.; and Judd, M. 2007. OASIS: Onboard Autonomous Science Investigation System for Opportunistic Rover Science. *Journal of Field Robotics* 24(5):379–397.

[Chien et al. 2004] Chien, S.; Sherwood, R.; Tran, D.; Cichy, B.; Rabideau, G.; Castano, R.; Davies, A.; Lee, R.; Mandl, D.; Frye, S.; Trout, B.; Hengemihle, J.; D'Agostino, J.; Shulman, S.; Ungar, S.; Brakke, T.; Boyer, D.; VanGaasbeck, J.; Greeley, R.; Doggett, T.; Baker, V.; Dohm, J.; and Ip, F. 2004. The EO-1 Autonomous Science Agent. In *Proc. of the 3rd Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-04)*, 420–427.

[Chong, Givan, and Chang 2000] Chong, E.; Givan, R.; and Chang, H. 2000. A Framework for Simulation-based Network Control via Hindsight Optimization. In *Proc. of the IEEE Conference on Decision and Control (CDC-00)*.

[Dean, Goemans, and Vondrak 2004] Dean, B.; Goemans, M.; and Vondrak, J. 2004. The Benefit of Adaptivity: Approximating the Stochastic Knapsack Problem. In *Proc. of the 45th annual IEEE Symposium on the Foundations of Computer Science (FOCS-04)*, 208–217.

[Garcia, Prett, and Morari 1989] Garcia, C.; Prett, D.; and Morari, M. 1989. Model Predictive Control: Theory and Practice–A Survey. *Automatica* 25(3):335–348.

[Hentenryck and Bent 2006] Hentenryck, P. V., and Bent, R. 2006. *Online Stochastic Combinatorial Optimization*. MIT Press.

[Kellerer, Pferschy, and Pisinger 2004] Kellerer, H.; Pferschy, U.; and Pisinger, D. 2004. *Knapsack Problems*. Springer.

[Puterman 1994] Puterman, M. 1994. *Markov Decision Processes, Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

[Teichteil-Königsbuch and Infantes 2009] Teichteil-Königsbuch, F., and Infantes, G. 2009. Anytime Planning in Hybrid Domains using Regression, Forward Sampling and Local Backups. In *Proc. of the the 4th ICAPS Workshop on "Planning and Plan Execution for Real-World Systems" (ICAPS-09)*.

[Thompson, Smith, and Wettergreen 2008] Thompson, D.; Smith, T.; and Wettergreen, D. 2008. Information-Optimal Selective Data Return for Autonomous Rover Traverse Science and Survey. In *Proc. of IEEE International Conference on Robotics and Automation (ICRA-08)*, 968–973.

[Woods et al. 2009] Woods, M.; Shaw, A.; Barnes, D.; Price, D.; Long, D.; and Pullan, D. 2009. Autonomous Science for an ExoMars Rover-Like Mission. *Journal of Field Robotics* 26(4):358–390.