A Local Search Solution for the INTEGRAL Long Term Planning

Matthew Kitching and **Nicola Policella**
European Space Agency, Darmstadt, Germany
name.lastname@esa.int

Abstract

The ESA INTEGRAL mission aims at observing gamma-ray emissions from the universe using several scientific instruments. The unique functionality of this spacecraft results in significant interest from the scientific community. Each year, the mission management team at ESA posts an announcement of opportunity (AO) to which scientists can submit observation requests. This results in an over-constrained scheduling problem where the objective is to perform each observation as much and as well as possible, while respecting the different spacecraft, flight dynamic, and observations constraints/preferences.

This work discusses the use of a local search approach based on a novel model of the original problem. This led to a search procedure more focused on the objectives of problem while keeping an high level coverage of the search space.

Introduction

The goal of the ESA INTEGRAL observatory is to gather the most energetic radiation in the Universe: Gamma rays. Gamma radiation from space cannot be tracked from the ground because it is blocked by Earths atmosphere. That is why INTEGRAL has to be a satellite-based observatory.[1] The satellite uses two specially designed gamma-ray telescopes to register these elusive rays. One telescope (*IBIS*) takes pictures of the gamma rays and the other measures their energy (*SPI*). The gamma-ray telescopes are supported by an X-ray monitor (*Jem-X*) and an optical camera (*OMC*). All four instruments point to the same region of the sky and take observations simultaneously.

The unique functionality of this spacecraft results in significant interest from the scientific community. Each year, the mission management team at ESA posts an announcement of opportunity (AO) to which scientists can submit observation requests. This results in an over-constrained scheduling problem where the objective is to perform each observation as much and as well as possible, while respecting the different spacecraft, flight dynamic, and observations constraints/preferences.

(Pralet and Verfaillie 2009) first formalized this problem and provided a solution based on local search. The specific algorithm was developed on top of the APSI planning framework (Cesta and Fratini 2008) and was integrated into the AIMS tool. This initial tool has been integrated with the local infrastructure and it is now operational in the ESA INTEGRAL Science Planning group.

This work introduces an alternative solution based on a novel model of the original problem. Two algorithms are introduced which both modify the variables of the problem in order to dynamically and efficiently change the neighborhoods explored in the local search. Varying the neighborhoods allows a large coverage of the search space, while still allowing focused exploration of the search space near local optima.

The paper is organized as it follows: first we recall the Integral problem, and describe a novel solving approach. Then we empirically evaluate this new solution against the one proposed by (Pralet and Verfaillie 2009) in particular. We conclude by analyzing the current solution and how this can be further improved and generalized.

Related works

In the past decades, the space domain has been a fertile area to efficiently apply automated planning and scheduling solutions. The problem presented in this paper is an oversubscribed problem, i.e., a problem in which the resources available (e.g., time, capacity) are not sufficient to accomplish all stated tasks or goals. The basic objective is to maximize the number of tasks accommodated or goals satisfied. Similar problems can arise in satellite domains such as the ones described in (Bensana, Lemaitre, and Verfaillie 1999; Verfaillie and Lemaitre 2001). Both works concern a set of Earth observation operations to be allocated over time under a set of mandatory constraints such as: no overlapping images, sufficient transition times (or *setup* times), bounded instantaneous data flow, and limited on-board recording capacity. In (Knight 2006) the author instead focuses his attention on the case of swath segment (i.e., fixed trajectory observations) scheduling for orbiting spacecraft, investigating different algorithms to obtain optimal solutions. Other examples can be found in different areas from rover task (Smith 2004) to military airlift allocation (Kramer and Smith 2004).

The technical solutions presented in this paper are based on local search (Van Hentenryck and Michel 2005; Aarts

[1] More details can be found at the mission overview website, http://www.esa.int/science/integral

and Lenstra 2003), in particular, the case of oversubscribed scheduling problems, as seen in (Kramer, Barbulescu, and Smith 2007). For oversubscribed problems, two basic classes of techniques have emerged: "searching directly in the space of possible schedules and searching in an alternative space of task permutations." The approach presented here directly searches the space of feasible solutions.

The Integral Problem

As previously mentioned the unique functionality of INTEGRAL raises many different requests from the science community, each of them focused on a particular scientific target. However it has to be considered that the satellite is moving on a highly elliptical orbit around the Earth. One revolution takes 72 hours, of which only 58 hours, the time not in the Earth radiation belts, are available for observations. The presence of the Sun, the Earth, the Moon, and other planets, makes a given target only intermittently observable during those 58 hours. For each target and each satellite revolution, observation windows can be pre-computed.

The observation of a given target requires a set number of observation patterns to be performed. There are four classes of patterns: rectangular patterns requiring 25 pointings, hexagonal patterns requiring 7 pointings, staring patterns requiring only one pointing, and user-defined patterns. Each pointing has a predefined duration. However it is not mandatory (and often impossible) to perform an observation within a single revolution: an observation can be divided into several sub-observations, each one including a chosen number of patterns. Also, a single observation pattern can be divided into sub-patterns, each one including a set number of pointings.

Several different types of observations are needed. Normal observations (NO) should be split as little as possible and ended as early as possible after they started; no-splitting observations (NS) should not be interleaved with other observations; periodic observations (PE(p,t)) should be decomposed into elementary observations to be performed every p revolutions with a tolerance of t on the deviation from the period; spread observations (SP(n)) should be decomposed into n sub-observations to be spread as much as possible over the year, and each sub-observation should be performed with no splitting.

Having given an high level description of the INTEGRAL context we can define a scheduling problem as follows:

- A set of non-overlapping revolutions, R. For each **revolution**, $r \in R$, a starting time, $S(r)$, and an ending time, $E(r)$, of the window available for the observations within r, as well as the maximum filling percentage $M(r)$ of r. The duration of r is $E(r) = S(r)$.
- A set of **priority weights**, P, where for each priority weight, $p \in P$, the value of p reflects the relative importance of observations of priority p.
- A set of **observations**, O. For each observation, $o \in O$, the observation has a type $TY(o)$ (NO for a normal observation, NS for no-splitting observation; (PE(p,t)) for a periodic observation to be performed every p revolutions with a tolerance of t on the deviation from the period, or SP(n) for spread observations which must be decomposed into n sub-observations to be spread as much as possible over the year and to be each performed with no splitting), a priority level $Priority(o) \in P$, a total Duration $D(o)$, the duration $DEO(o)$ of an elementary observation, a number $NEO(o)$ of elementary observations ($D(o) = NEO(o) \cdot DEO(o)$), and a percentage $PCA(o)$ above which o is considered to be achieved.
- A set $W(o)$ of **observation windows** available for o. For each observation window $w \in W(o)$, a revolution $R(w) \in R$, a starting time $S(w)$ and an ending time $E(w)$; several windows may be available for performing an observation o within a revolution r; however, they do not overlap; if $TY(o) = SP(n)$, observation o is decomposed into a set $SO(o)$ of n sub-observations and the set $W(o)$ of the windows associated with o is partitioned into n subsets, each set $W(so)$ is associated with a sub-observation $so \in SO(o)$. Each revolution, $r \in R$, has a set of observation windows which includes all windows w where $R(w) = r$.
- MOA is the maximum number active observations per revolution, useful to limit time spent on slewing between observation activities.
- The output is a set AWS of **active window segments**. For each active window $aws \in AWS$, there is a revolution $R(aws) \in R$, an observation $O(aws) \in O$, a window $W(aws) \in W(O(aws))$, a starting time $S(aws)$, where $S(aws) \geq S(W(aws))$ and an ending time $E(aws)$, where $E(aws) \leq E(W(aws))$ and $E(aws) - S(aws)$ is a multiple of $DEO(O(aws))$. An observation, o, is **active** in revolution, r, if there is some active window segment $aws \in AWS$, with $R(aws) = r$ and $O(aws) = o$. A feasible solution (plan) for the Integral Problems will consist of a set of non-overlapping active window segments such that the number of active observations is less than MOA. In addition, the active window segments must satisfy a set the constraints listed later in this paper.

The Model

The approach presented in this paper is based on a transformed model of the problem: this is based on the assumption that each revolution (or sub-revolution described below) has at most a single active observation. After solving the transformed problem, the solution is mapped back to the original specification of the problem.

In order to model the problem in this way, the following ideas are introduced. First, the algorithm maintains a set of **sub-revolutions**, R^s, where each $r^s \in R^s$ defines a sub-section of an original revolutions, $r \in R$. For each sub-revolution, $r^s \in R^s$ there is a parent revolution, $Parent(r^s) \in R$, a starting time, $S(r^s)$ where $S(r^s) \geq S(r)$, an ending time, $E(r^s) where E(r^s) \leq E(r)$, the maximum duration available for observation during the revolution, $MaxFill(r^s)$. Finally, each original revolution, $r \in R$ has $Children(r)$, which includes those sub-revolutions, r^s, where $Parent(r^s) = r$.

The algorithms presented in this paper segments the original revolutions of the problem into sub-revolutions and considers plans with only a single observation for every sub-revolution. Although this imposes an additional constraint on the problem, it allows the solver to make simpler, faster moves, thereby exploring more of the search space. The previous solution to this problem called a scheduler after each local move in order to know whether or not the set of active window segments present in a revolution are schedulable. By assuming only a single observation for each sub-revolution, there is no need to call a scheduler since schedulability is guaranteed after any local move.

Let **State**(r^s) be the state of r^s, where $State(r^s) \in O \bigcup \{emptyObs\}$, where **emptyObs** is the empty observation. If $State(r^s) = o$, then the satellite can observe only observation o during the time period from $S(r^s)$ to $E(r^s)$ (in the case of State$(r^s) = emptyObs$, the satellite is not making any observations during the time period).

Revolution $r^s \in R^s$ has an overlap with each observation window, $w \in W(o)$. Function **overlap**(r^s, w) returns the size of the intersection between the start and end points of the revolution and the start and end points of the observation window rounded down to the nearest multiple of DEO(o). Formally, if $minEnd = min(E(r^s) - E(w))$ and $maxStart = max(S(r^s), S(w))$, then $overlap(r^s, w)$ returns:

$$\lfloor \frac{max(0, minEnd - maxStart)}{DEO(o)} \rfloor \cdot DEO(o) \quad (1)$$

There is one exception to procedure overlap. When $w \in W(o)$ and TY(o) = PE(p,t), then $overlap(r^s, w)$ returns at most a duration of DEO(o). Periodic observations must not have more than a single elementary observation per window.

For any plan, **CurObsDur**(r^s, o), is the duration that the satellite is observing o during r^s. The value of $CurObsDur(r^s, o)$ must be a multiple of DEO(o). **TotalObsDur**(o) is the total duration that the satellite is observing observation $o \in O$, equal to $\sum_{r^s \in R^s} CurObsDur(r^s, o)$. Thus, TotalObsDur(o) is also a multiple of DEO(o).

The algorithms presented in this paper simply returns a set of sub-revolutions, R^s, states for the revolutions, and durations $CurObsDur(r^s, o)$. An exact plan can be extracted from this information by calls to a procedure $extractPlanforRevolution(r^s)$, seen in Algorithm 1. This procedure returns a set of active window segments, AWS, by creating active window segments based on the observation windows of the original problem specification.

Figure 1 illustrates how the original problem is transformed into the model, and how plans are then extracted from the transformed problem. Figure 1.a) shows a small Integral Problem instance, with two observations, one revolutions, and six observation windows. Figure 1.b) shows the same problem with r_1 split into two sub-revolutions, r_1^s and r_2^s. The empty rectangles show $overlap(r_1^s, o_1)$, $overlap(r_2^s, o_1)$, $overlap(r_1^s, o_2)$, and $overlap(r_2^s, o_2)$. Note that the problem is a transformation of the initial problem; start and end points of observation windows are replaced by simple maximum durations for each observation.

Algorithm 1 extractPlanforRevolution(r^s)

```
o = State(r^s)
if o = emptyObs then
  return ∅ ;
end if
AWS = ∅;
for all w ∈ W(o) do
  dur_w = min(overlap(r^s, w), CurObsDur(r^s, o));
  if dur_w > 0 then
    aws = new active window segment;
    S(aws) = max(S(w), S(r^s));
    E(aws) = S(aws) + dur_w;
    R(aws) = Parent(r^s);
    O(aws) = o;
    AWS = AWS ⋃ aws;
    CurObsDur(r^s, o) = CurObsDur(r^s, o) − dur_w;
  end if
end for
return AWS;
```

Figure 1.c) shows the transformed problem where solid black rectangles represent that $CurObsDur(r_1^s, o_1)$ = $overlap(r_1^s, o_1)$ and $CurObsDur(r_2^s, o_2)$ = $overlap(r_2^s, o_2)$. Figure 1.d) represents this plan mapped back to the original definition of the Integral Problem after calls to $extractPlanforRevolution(r_1^s)$ and $extractPlanforRevolution(r_2^s)$. Here, solid black rectangles represent active window segments (meaning the satellite is observing an observation during the timeframe).

Given that revolutions are non-overlapping, and the sub-revolutions of any revolution are non-overlapping, then when extractPlanforRevolution is called with any current plan, the active window segments produced are also non-overlapping.

Before concluding this section, it is worth remarking on the differences between the approach here presented and the work described in (Pralet and Verfaillie 2009). The latter allows many different active observation segments for each revolution. However, during the local search, the algorithm must check after every move whether the observations during a window are schedulable. The model presented here does not allow different observations to be scheduled during a single sub-revolution. As we will see in the algorithm section of this paper, this allows fast local moves to be made.

Constraints

Both the constraints and Criteria sections are based on those proposed in (Pralet and Verfaillie 2009), with modifications made to fit into our model. A feasible plan must satisfy six different types of constraints. In order to define the constraints, the following are defined:

- Let $seqObs(o)$ be the sequence of revolutions where State(r) = o, ordered according to their starting time, where $nextObs(r^s)$ the next observation in $seqObs(o)$.

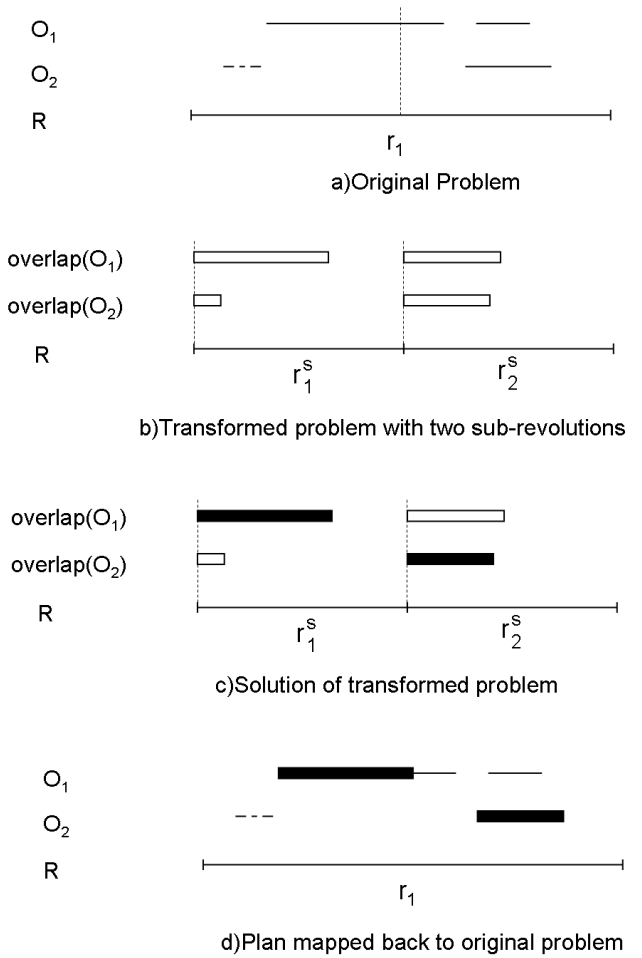

Figure 1: Integral Problem

- Let $foa(o)$ (resp. $loa(o)$) be the first (resp. last) revolution in seqObs(o) (equal to $\emptyset$ if seqObs(o) = $\emptyset$).

The six types of constraints are then:

1. For each revolution, r, the total current duration of the sub-revolutions of Children(r) is less than or equal to $((E(r) - S(r)) \cdot M(r)$. Thus, $\forall r \in R, \sum_{r^s \in Children(r)} CurObsDur(r^s, o) \leq ((E(r) - S(r)) \cdot M(r)$.
2. For each revolution, r, the maximum number of active observations is less than or equal to MOA. Each sub-revolution in $Children(r)$ can have only a single active observation. Thus, $\forall r \in R, |Children(r)| \leq MOA$.
3. For each no-splitting observation o, o must not be interleaved with other observations. Thus, $\forall o \in O|(TY(o) = NS)$, if R' is the set of sub-revolutions between foa(o) and loa(o), then $\forall r^s \in R', State(r^s) = o$.
4. For each periodic observation o, only a single elementary observation per revolution. Thus, $\forall o \in O|(TY(o) = PE(p,t))$, if R' is the set of sub-revolutions in state o, then $\forall r^s \in R', (\textbf{CurObsDur}(r^s, o) = DEO(o))$.
5. For each periodic observation o of periodicity p and of tolerance t, active observations of o must be separated by distance p, within a tolerance of t. Thus, $\forall o \in O|(TY(o) = PE(p,t))$, if R' is the set of sub-revolutions in state o, then $\forall r^s \in R', r^s = loa(o) \vee |Parent(nextObs(r^s)) - Parent(r^s) - p| \leq t$
6. For each spread observation, o, the maximum duration of active window segments associated with each sub-observation: $\forall o \in O|(TY(o) = SP(n)), (\sum_{r^s \in R^s} CurObsDur(r^s, o)) \leq D(o)/n$;

Criteria

For each observation o, let $q(o)$ be the quality associated with o in the current plan. The global criterion q to be maximized is defined as the normalized weighted sum of observation qualities:

$$q = \frac{\sum_{o \in O} Priority(o) \cdot q(0)}{\sum_{o \in O} Priority(o)} \tag{2}$$

For each observation o, let qc(o) be the completion quality of o and qr(o) be its realization quality. Let $\alpha \in [0,1]$ be a user defined constant which expresses the trade-off between observation completion and realization. The quality q(o) associated with o is defined as the weighted sum of completion and realization qualities of o:

$$\forall o \in O : q(o) = \alpha \cdot qc(o) + (1 - \alpha) \cdot qr(o) \tag{3}$$

For each observation, o, the completion quality, qc(o) depends on the percentage completion of o:

$$qc(o) = min(1, \frac{TotalObsDur(o)}{D(o) \cdot PCA(o)}) \tag{4}$$

The realization quality qr(o) of an observation depends on the type of observation. For each normal or no-splitting observation o, the objective is to finish o as early as possible after it started. Thus, the realization quality depends on the distance (in terms of revolutions) between the last and the first active observation activity associated with o. Let $\delta(o) = loa(o) - foa(o)$ be this distance. Let $\Delta^{max}(o)$ be the maximum value of $\delta(o)$, obtained when the first and last revolutions of o are both active. Let $\Delta^{min}(o)$ be the minimum value of $\delta(o)$ which could be obtained if the plan would contain only elementary observations associated with o. Both $\Delta^{max}(o)$ and $\Delta^{min}(o)$ can be pre-computed for each possible value of TotalObsDur(o). Thus:

$$\forall o \in O|TY(o) \in \{NO, NS\} :$$
$$qr(o) = \begin{cases} 0 \; if(TotalObsDur(o) = 0) \\ 1 \; if(\Delta^{min}(o) = \Delta^{max}(o)) \\ \frac{\Delta^{max}(o) - \delta(o)}{\Delta^{max}(o) - \Delta^{min}(o)} \; otherwise \end{cases} \tag{5}$$

For each periodic observation o, the objective is to satisfy as well as possible the periodicity constraint. Thus, the realization quality depends on the sum of the deviations from the period. Let

$sumdev(o) = \sum_{r^s \in R^s | r^s < loa(o)} |Parent(nextObs(r^s)) - Parent(r^s) - p|$ be this sum. Let $maxdev = (TotalObsDur(o)/DEO(o) - 1) \cdot t$ be the maximum deviation possible. Thus:

$$\forall o \in O | TY(o) = PE(p,t) :$$
$$qr(o) = \begin{cases} 0 \; if(TotalObsDur(o) = 0) \\ 1 \; if(TotalObsDur(o) = DEO(o)) \\ 1 - \frac{sumdev(o)}{maxdev(o)} \; otherwise \end{cases} \quad (6)$$

Finally, for each spread observation o, its realization quality is defined as the mean value of the realization qualities of its associated no-splitting sub-observations:

$$\forall o \in O | TY(o) = SP(n) :$$
$$qr(o) = \frac{\sum_{so \in SO(o)} qr(so)}{n} \quad (7)$$

The Algorithm

The algorithms described in this paper are local search algorithms that operate in the feasible space. This section begins with a description of some of the core procedures used in the algorithms. Then, a detailed description of the first algorithm, Coarse Grained Local Search, CGLS is given. The second part of the section is instead dedicated to the second algorithm, Refined Local Search, RLS.

Function **move**(r^s, o_{new}) changes $State(r^s) = o_{old}$ to $State(r^s) = o_{new}$. $CurObsDur(r^s, o_{old})$ is set to zero and $CurObsDur(r^s, o_{new})$ is set to the maximum possible duration that can be assigned to observation o_{new} given the current state of the plan. The maximum duration is the minimum of: 1)The maximum duration allowed by the revolution, 2)The maximum duration allowed by the observation (given that D(o_{new}) cannot be exceeded), and 3) The maximum overlap between windows of the observation, and the revolution. Formally:

$$\textbf{Avail}(r^s, o) = min \begin{cases} \lfloor \frac{MaxFill(r^s)}{DEO(o)} \rfloor \cdot DEO(o) \\ D(o) - TotalObsDur(o) \\ \sum_{w \in W(o)} overlap(r^s, w) \end{cases} \quad (8)$$

Central to the algorithm is the **GetDeltaChange** procedure, described in Algorithm 2, which takes as parameters a sub-revolution, r^s and the new state of the revolution, o_{new}, and returns the change in criteria that would occur if $move(r^s, o_{new})$ was made. The procedure returns a change in criteria of $-\infty$ if $move(r^s, o_{new})$ would violate a constraint or is tabu. Next, the procedure calculates the change in criteria, $\Delta^-_{r^s, o_{old}}$, that occurs when a duration of $CurObsDur(r^s, o_{old})$ is removed from observation o_{old}. $CriteriaChange$ evaluates the difference between equations 2-7 in the Criteria section before and after decreasing $CurObsDur(r^s, o_{old})$ to zero. Next, the procedure calculates the change in criteria when the maximum possible duration is added to o_{new} by evaluating equations 2-7. The procedure returns the total change in criteria that would occur if $State(r^s)$ was changed to o_{new}.

Algorithm 2 $GetDeltaChange(r^s, o_{new})$

$o_{old} = State(r^s)$;
if $move(r^s, o_{new})$ is infeasible or tabu **then**
  return $-\infty$;
end if
$\Delta^-_{r^s, o_{old}} = CriteriaChange(o_{old}, -CurObsDur(r^s, o_{old}))$;
$\Delta^+_{r^s, o_{new}} = CriteriaChange(o_{new}, Avail(r^s, o_{new}))$;
return $\Delta^+_{r^s, o_{new}} + \Delta^-_{r^s, o_{old}}$;

Next, a procedure **CreateRev(start, end, maxFill, observation)** is defined which returns a newly created revolution, r^s, with $S(r^s) = start$, $E(r^s) = end$, and $MaxFill(r^s)$ = maxFill, and $move(r^s, observation)$ is made.

Given a revolution, $r \in R$, r can be partitioned into sub-revolutions by making a call to Split(see Algorithm 3) with the parameter numberSegments, which specifies the number of segments. The procedure splits a revolution into numberSegments sub-revolutions, where the duration and MaxFill of the original revolution is split evenly among the sub-revolutions.

Algorithm 3 $Split(r, numberSegments)$

segLength = $(E(r) - S(r))$/numberSegments;
maxFill = $((E(r) - S(r)) \cdot M(r))/(numberSegments)$;
Revs = $\emptyset$;
for segment from 1 to numberSegments **do**
  s = $S(r) + (segment - 1) \cdot segLength$;
  e = $S(r) + segment \cdot segLength$;
  newRev = CreateRev(s, e, maxFill, emptyObs);
  $Parent(newRev) = r$;
  Revs = Revs $\bigcup$ newRev;
end for
$Children(r)$ = Revs;
return Revs;

The first local search algorithm introduced in this paper is the Coarse Grained Local Search procedure (see Algorithm 4) which is a standard tabu local search. The input to the problem is a set of sub-revolutions, $R^s = R$. Thus, for this algorithm, each revolutions of the original problem has a single sub-revolutions in R^s.

The tabu list consists of moves. If a move from $State(r^s) = o_{old}$ to $State(r^s) = o_{new}$ is in the tabu list, a move back to $State(r^s) = o_{old}$ is tabu. The length of the tabu list is equal to the number of observations in the problem.

The variable Criteria represents the score of the current plan, and bestLocalCriteria represents the best score achieved during this iteration of the local search. The local search continues to make moves until the score has not improved from the best local score in β number of moves.

At each iteration of the local search, the best move is the move that maximizes the change in criteria, and is found by evaluating $GetDeltaChange$ for all possible moves. The criteria is then updated, and the best move is made. Although the change in Criteria may be negative, any move

will remain in feasible space due to the check for $-\infty$.

It is important to note that the values $\Delta^+_{r^s,o_{old}}$ and $\Delta^-_{r^s,o_{old}}$ used in $GetDeltaChange$ do not necessarily change between every move of the local search. If a move is made from $State(r^s) = o_{old}$ to $State(r^s) = o_{new}$ then the value of $\Delta^-_{r^s,o}$ does not change if $o \neq o_{old}$ and $o \neq o_{new}$. Likewise, $\Delta^+_{r^s,o}$ does not change if $o \neq o_{old} \neq o_{new}$. Thus, the algorithm can maintain scores for $\Delta^+_{r^s,o}$ and $\Delta^-_{r^s,o}$ for every pair of revolutions and observations. When a move is made, $UpdateDeltasOfObservation(o_{old})$ (resp. $UpdateDeltasOfObservation(o_{new})$ updates all the values of $\Delta^-_{r^s,o_{old}}$ and $\Delta^-_{r^s,o_{old}}$ for every $r^s \in R^s$. Thus, a significant amount of work is avoided by remember previous values of $\Delta^-_{r^s,o_{old}}$ and $\Delta^-_{r^s,o_{old}}$.

Finally, the values of indexWithoutImprovement and bestLocalCriteria are updated based on whether Criteria is greater than bestLocalCriteria.

Algorithm 4 CGLS(R^s)

```
Criteria= 0;
bestLocalCriteria = 0;
indexWithoutImprovement = 0;
while indexWithoutImprovement < β do
  (r^s, o_new) = max_{∀r^s∈R^s, o∈O} GetDeltaChange(r^s, o);
  if GetDeltaChange(r^s, o) = −∞ then
    continue;
  end if
  Criteria += GetDeltaChange(r^s, o_new);
  move(r^s, o_new);
  for all r^s ∈ R^s do
    UpdateDeltasOfObservation(o_old, r^s);
    UpdateDeltasOfObservation(o_new, r^s);
  end for
  if Criteria > bestLocalCriteria then
    indexWithoutImprovement = 0;
    bestLocalCriteria = Criteria;
    storeSolution();
  else
    indexWithoutImprovement++;
  end if
end while
```

Since each revolutions of the original problem has a single sub-revolutions in R^s, this procedure simply tries assigning a single observation to each revolution in the original problem. Thus, the complexity of the problem is greatly reduced, and moves can be made extremely quickly. However, as will be seen in the experimental section, the results of this algorithm are poor since limiting the plan to a single observation for each revolution restricts the search space too much.

This completes the basic local search algorithm. As a sequel, this paper introduces an extension of CGLS which refines the solutions found in Algorithm 4. The Refined Local Search, RLS is described in Algorithm 5. The search proceeds with a call to CGLS. If the solution produced by CGLS is better than the current best solution, the solution found is restored, and the algorithm attempts to fine tune this solution. This is done by segmenting each revolution, $r^s \in R^s$ based on the restored solution. First, a sub-plan is extracted for each revolution, by calling extractPlanforRevolution, which returns a set of active observation segments. The minimum end time and maximum start time is then found over all the active window segments.

For each revolution $r^s \in R^s$, a call to splitRefine (Algorithm 6) is made, and each revolution is split into three segments; 1) the first revolution before the minimum start time, 2) the last revolution after the maximum end time, and 3) the middle revolution in between these two points. The state of the first and last revolutions is set to $emptyObs$, but the state of the middle revolution is set to the current state of r^s. Note that splitRefine creates revolutions not only with different durations and states, but also different values of MaxFill. The MaxFill of the middle revolution is set to the entire duration of the revolution, since the entire duration of the revolution is used by the observation that is currently being observed by the middle revolution. The remaining MaxFill of the original revolution is then split between the first and last revolution based on the respective size of the first and last revolution. Effectively, splitRefine is finding segments at the beginning and end of each revolution that are not being used in the current plan, and creating new empty revolutions that can be used to make other observations.

RLS is then recursively called with an increased number of revolutions in R^s. The depth of this recursion is bounded by the fact that bestLocalCriteria is monotonically increasing and is bounded by one. In practice, the depth of the recursion was never greater than 4 on the experiments conducted.

Algorithm 5 RLS(R^s, GBS)

```
Score = 0;
bestLocalCriteria = CGLS(R^s);
if bestLocalCriteria > GBS then
  GBS = bestLocalCriteria;
  restoreSolution();
  for all r^s ∈ R^s do
    if |children(P(r^s))| ≤ MOA − 2 then
      R^s = R^s \ r;
      AWS = extractPlanforRevolution(r^s);
      minTime = min_{w∈AWS} S(w)
      maxTime = max_{w∈AWS} E(w)
      [Rev] = splitRefine(r^s, minTime, maxTime);
      R^s = R^s ⋃ Rev;
    end if
  end for
  RLS(R^s, GBS);
end if
```

Figure 2 illustrates how RLS modifies the problem. Suppose Figure 2.a) represents a solution for the two revolutions seen in Figure 1, where the dark rectangles are active window segments in the plan. Consider revolution r^s_2 in Figure 2.a). The revolution will be split into three sub-revolutions; 1) revolution r^s_3 before the minimum of the observation windows, and 2) revolution r^s_5 after the maximum of the observation windows, and 3) revolution r^s_4 between these two revolutions (see Figure 2.b). As seen in Fig-

Algorithm 6 $splitRefine(r^s, s, e)$

$mid = createRev(s, e, (end - start), State(r^s))$;
$remFill = MaxFill(r^s) - (end - start)$;
$remDur = (E(r^s) - S(r^s)) - (E(mid) - S(mid))$;
$fillLower = remFill \cdot (start - S(r^s))/remDur$;
$fillUpper = remFill \cdot (E(r^s) - end)/remDur$;
$lower = createRev(S(r^s), s, fillLower, emptyObs)$;
$upper = createRev(e, E(r^s), fillUpper, emptyObs)$;
$newRevs = mid \bigcup lower \bigcup upper$;
$Parent(mid, lower, upper) = Parent(r^s)$;
$Children(P(r)) = Children(P(r)) \backslash r \bigcup newRevs$;
return $newRevs$;

ure 2.c), the middle window, r_4^s, will remain in state Obs 2 and the other windows are set to $emptyObs$. In this case, the split allows duration to be placed into $CurObsDur(r_3^s, o_1)$ during the recursive call to RLS.

Computational Advantages

The first immediate advantage of the proposed model and algorithms is a reduction of the complexity of the problem. By assuming that each sub-revolution has no more than one active observation, there is a dramatic reduction in the complexity of the problem. Furthermore, by placing the maximum allowable duration in each sub-revolution, the local search does not need to calculate the schedulability of many observations within a revolution.

A second computational benefit comes from the fact that the first two constraints presented in the constraint section do not need to be checked when a move is made; they are true for all possible moves that are not tabu. Thus, these constraints can be ignored when calculating whether any potential move is feasible.

Experimental Results

Our local search approach was tested on real Integral Problem instances. All tests were performed on 3.16 GHz machines with 3.5GB of RAM and limited to a timeout of 600 seconds.

We have implemented the algorithms described above, and tested them on three sets of benchmarks; each benchmark consisting of between 12 and 18 months of data from the **Integral** mission. For each benchmark, α (used in Equation 3) was varied between 0 and 1 at increments of 0.05.

Specifically, the following three algorithms were tested. (1) **aims**, which a tabu based local search algorithm, which starts from an empty plan and uses local moves which add and remove active window segments with flexibility to choose any possible start and end point for the active window segment. The algorithm uses a stochastic heuristics to select, to choose, and to accept local moves, a tabu list to avoid cycles in the sequence of local moves, and restarts to diversify the search. (2) **cgls**, which is the local search algorithm introduced in Algorithm 4. (3) **rls**, which has been introduced in Algorithm 5.

Figure 3 shows the results of the three algorithms on the Integral benchmarks with varying values of α. When α is

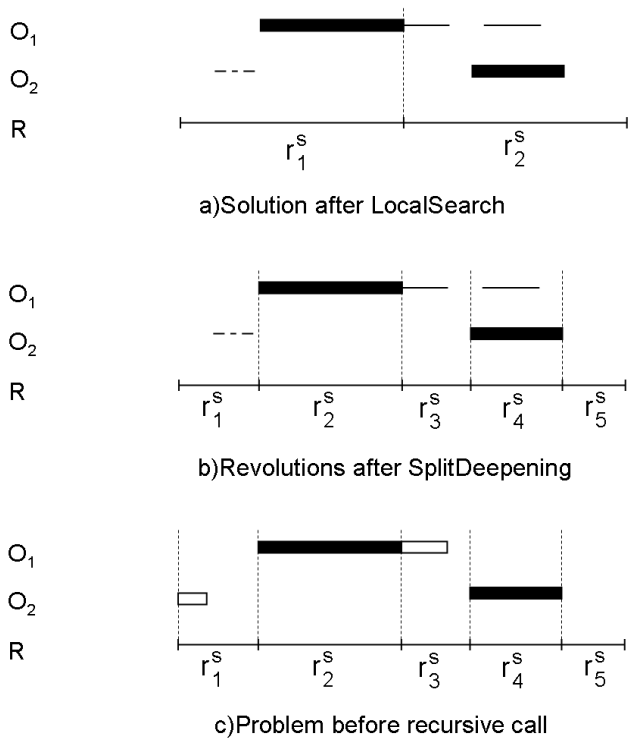

Figure 2: Refined Local Search Example

one, the criterion is entirely determined by the percentage completion of observations. The original AIMS algorithm is able to outperform CGLS on two of the three benchmark sets. This is due to the fact that AIMS is able to schedule window segments representing many different active observations in a single revolution, whereas CGLS is limited to a single active observation for each revolution.

The results also show RLS outperforming the other two algorithms on almost all instances. RLS is able to make simpler and much faster moves, and is thus able to find better global solutions. In the Integral mission, the default value for MOA, which limits the number of observations per revolution in any solution, is five. By recursively splitting revolutions, RLS is able to schedule five observations during each original revolution at a depth of three. Thus, although it may appear that restricting the plan to one observation per sub-revolution is constraining, RLS is able to schedule the maximum allowed number of observations per revolution for this mission.

Table 1 gives the detailed results of the AIMS and RLS algorithms on all three sets of benchmarks. The first column of each algorithm shows the best criteria found in the timeout, the time in seconds to find the best plan, the number of moves to find the best plan, and the total number of moves. This table confirms that RLS is able to make a far greater number of moves, in many cases making over two orders of magnitude more moves within the timeout. The AIMS algorithm often converges quickly, and is then unable to improve on the solution. Conversely, the time and nodes to

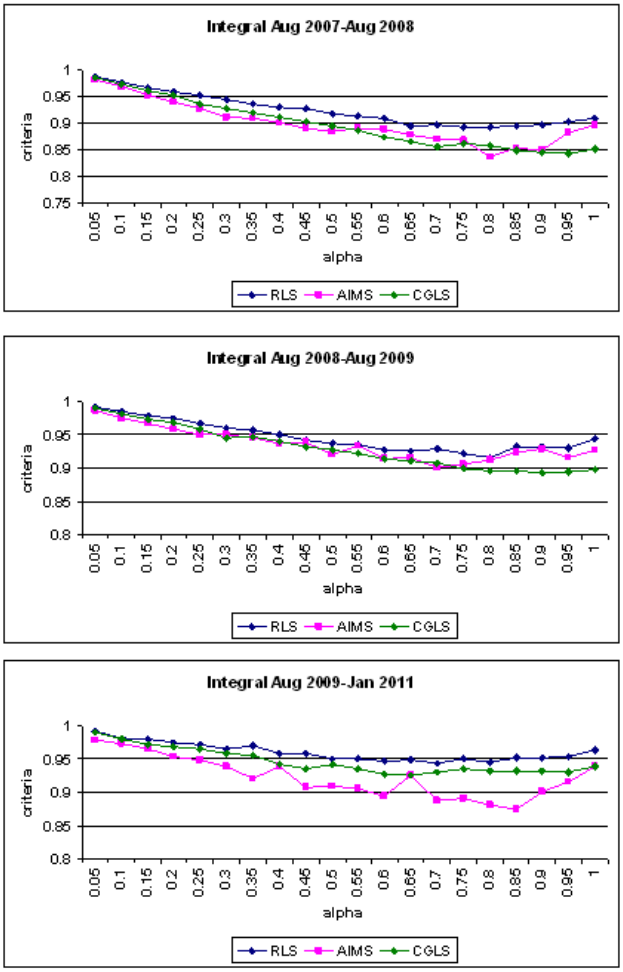

Figure 3: Criteria of Integral Problem with varying α

the best solution for RLS tend to be higher, showing that the algorithm continues to improve on solutions as local search continues.

Conclusions

This paper presents two alternative algorithms that solve an oversubscribed problem identified in the domain of the ESA INTEGRAL mission. Both are local search algorithms which work on different level of granularity of the basic moves. The first having a coarse grained view of the problem which allows a fast, wide coverage of the search space. The second algorithm extends the previous approach by refining the neighborhood of the local optima obtained in the first case.

The empirical evaluation has shown the efficiency of these approaches compared to the previous solution described in (Pralet and Verfaillie 2009). The efficiency of the algorithms results from the model of the problem allowing the algorithms to dynamically change the granularity of the neighborhood of the search space.

Future work will be focused on exploring the possible generalization of the techniques to other oversubscribed problems. Also, the possibility of extending the current approach will be considered.

Acknowledgment. The authors would like to thank Erik Kuulkers, space planning responsible for the INTEGRAL mission, for the support in the problem definition and to analyze and evaluate the solutions here presented.

References

Aarts, E., and Lenstra, J., eds. 2003. *Local Search in Combinatorial Optimization*. Princeton University Press.

Bensana, E.; Lemaitre, M.; and Verfaillie, G. 1999. Earth Observation Satellite Management. *Constraints: An International Journal* 4(3):293–299.

Cesta, A., and Fratini, S. 2008. The Timeline Representation Framework as a Planning and Scheduling Software Development Environment. In *In PlanSIG-08, Proceedings of the 27th Workshop of the UK Planning and Scheduling Special Interest Group*.

Knight, R. 2006. Solving Swath Problems Optimally. In *2nd IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT'06)*.

Kramer, L. A., and Smith, S. F. 2004. Task Swapping for Schedule Improvement: A Broader Analysis. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004), June 3-7 2004, Whistler, British Columbia, Canada*, 235–243.

Kramer, L. A.; Barbulescu, L.; and Smith, S. F. 2007. Understanding Performance Tradeoffs in Algorithms for Solving Oversubscribed Scheduling. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, 1019–1024.

Pralet, C., and Verfaillie, G. 2009. AIMS: A Tool for Long-term Planning of the ESA INTEGRAL Mission. In *Proceedings of the 6th International Workshop on Planning and Scheduling for Space, IWPSS09*.

Smith, D. E. 2004. Choosing Objectives in Over-Subscription Planning. In *Proceedings of the Fourteenth International Conference on Automated Planning and Scheduling (ICAPS 2004), June 3-7 2004, Whistler, British Columbia, Canada*, 393–401.

Van Hentenryck, P., and Michel, L. 2005. *Constraint-Based Local Search*. The MIT Press.

Verfaillie, G., and Lemaitre, M. 2001. Selecting and Scheduling Observations for Agile Satellites: Some Lessons from the Constraint Reasoning Community Point of View. In Walsh, T., ed., *Principles and Practice of Constraint Programming, 7th International Conference, CP 2001*, number 2239 in Lecture Notes in Computer Science, 670–684. Springer.

		AIMS				CGLS			
	α	criteria	time(s)	moves	tot moves	criteria	time(s)	moves	tot moves
Aug 2007 - Aug 2008	0.05	0.980	36	779	10374	0.987	137	409418	1809065
	0.1	0.969	17	407	10832	0.976	291	877380	1805168
	0.15	0.953	34	724	9507	0.967	401	1232826	1834236
	0.2	0.940	296	5400	10566	0.958	320	957837	1816913
	0.25	0.928	22	533	10672	0.952	240	709459	1780702
	0.3	0.910	24	560	9446	0.943	260	748599	1756653
	0.35	0.908	9	247	9271	0.937	554	1607093	1742578
	0.4	0.900	570	8548	9022	0.931	103	290217	1732104
	0.45	0.890	23	535	8863	0.929	209	587644	1686936
	0.5	0.885	10	261	8104	0.917	290	814988	1681854
	0.55	0.891	9	223	8550	0.914	478	1340851	1683651
	0.6	0.889	11	276	8049	0.909	596	1666308	1677424
	0.65	0.878	11	280	8474	0.894	557	1507202	1622963
	0.7	0.871	10	256	7509	0.896	581	1566782	1617889
	0.75	0.867	10	252	7627	0.893	391	1053861	1623308
	0.8	0.838	37	673	8082	0.893	392	1059510	1623684
	0.85	0.853	13	345	8278	0.894	596	1588230	1599273
	0.9	0.848	29	584	8443	0.897	188	486897	1601329
	0.95	0.882	11	290	7693	0.902	64	161846	1613731
	1	0.897	12	285	6996	0.909	550	1353455	1475340
Aug 2008 - Aug 2009	0.05	0.987	404	6255	9215	0.991	124	255119	1269645
	0.1	0.974	8	166	9609	0.985	462	960341	1253337
	0.15	0.967	530	8019	9008	0.979	282	588227	1256567
	0.2	0.959	38	779	9384	0.975	154	321575	1257544
	0.25	0.951	472	7092	8956	0.966	499	1038549	1252626
	0.3	0.950	224	3661	9271	0.961	284	589792	1248521
	0.35	0.946	9	190	8546	0.957	352	722301	1249407
	0.4	0.937	13	312	9114	0.950	499	1008349	1210706
	0.45	0.939	22	504	8576	0.943	362	734375	1225377
	0.5	0.920	53	998	8152	0.938	165	331422	1216004
	0.55	0.934	11	256	8798	0.936	29	48539	1221073
	0.6	0.914	400	5300	7946	0.928	463	916812	1192405
	0.65	0.915	469	6278	8010	0.925	154	299310	1152099
	0.7	0.901	37	745	8726	0.929	104	195515	1165200
	0.75	0.906	35	552	7726	0.922	424	801799	1136697
	0.8	0.912	12	286	7940	0.916	403	751574	1123653
	0.85	0.923	27	599	7647	0.932	454	845359	1118109
	0.9	0.929	14	324	7794	0.932	515	931800	1080655
	0.95	0.916	39	861	6601	0.930	308	549360	1069113
	1	0.927	11	242	6538	0.944	89	164315	1160033
Aug 2009 - Jan 2011	0.05	0.979	384	5876	6302	0.991	22	30136	1070536
	0.1	0.973	15	366	804	0.983	134	231476	1035268
	0.15	0.965	490	7049	8033	0.980	591	997219	1013825
	0.2	0.953	120	380	1699	0.975	350	591455	1023168
	0.25	0.949	71	246	1751	0.972	144	229124	991129
	0.3	0.939	142	563	2063	0.966	500	814057	984874
	0.35	0.921	196	914	2461	0.971	183	291508	969330
	0.4	0.940	239	1395	2936	0.959	313	487788	954223
	0.45	0.908	34	287	3750	0.959	270	427663	953629
	0.5	0.909	490	4900	5809	0.950	173	271986	972200
	0.55	0.906	424	897	1207	0.950	478	772847	968240
	0.6	0.894	221	562	1360	0.947	63	94297	977125
	0.65	0.925	74	188	1461	0.948	313	522864	1015802
	0.7	0.888	256	867	1803	0.944	423	717115	1026575
	0.75	0.891	328	1211	2013	0.951	519	917067	1063831
	0.8	0.882	329	1326	2132	0.946	251	449395	1066574
	0.85	0.875	137	771	2546	0.951	534	990280	1114996
	0.9	0.901	256	1441	3157	0.952	224	410497	1121714
	0.95	0.916	32	311	3816	0.954	430	804811	1129260
	1	0.939	45	628	4643	0.964	233	447829	1200151

Table 1: Best Criteria, Time to Best Criteria (sec), Moves to Best Criteria, and Total Move using 600 second timeout