

Coarse-grain Model of the SNAP Scheduling
Problem: Problem specification description and
Pseudo-boolean encoding

Geoff Pike, Donghan Kim and Alejandro Bugacov

July 11, 2003

1 Problem Specification

The coarse-grain model of the SNAP problem (a.k.a. ZMARBIES) is an extension to the original resource allocation MARBLES problem. To model the scheduling problem, in this new representation we include time. To encode the problem, we discretize time. The problem can be discretized at different resolutions with time-step ≥ 1 minute.

- A set $\mathbf{T} = T_1, T_2, \dots, T_N$ of N Tasks
- A set $\mathbf{R} = R_1, R_2, \dots, R_M$ of M Resources

1.1 Resources

Each Resource is described by:

- Name
- Type (E.g., Pilot, Aircraft, Simulator)
- A set of Availability Slot Ranges

An Availability Slot Range is a discretized time interval and is described by:

- Start slot: The first slot in the range
- Length: Number of slots in the range
- Capacity: Resource capacity during that time interval

There are two subclasses of resources:

- Reusable Resources: These resources can be used up to their capacity at each individual time slot.
- Consumable Resources: The usage of Consumable resources adds up and they can be used up to their capacity for the duration of the whole Availability Slot Range.

1.2 Tasks

Each Task is described by:

- Name
- Value
- A set of Start-Time Range Slots
- A set of Requirements

A Requirement is described by:

- Name
- Required Number
- Segment Length
- Offset from task start slot
- A set of Qualified Resources

2 Pseudo-Boolean Encoding

To encode the problem in Pseudoboolean form we introduce the following parameters and notation:

- N = Number of Tasks
- M = Number of Resources
- $\mathbf{R}_{reusable}$ = Set of Reusable Resources
- $\mathbf{R}_{consumable}$ = Set of Consumable Resources
- \mathbf{R} = Set of all resources. $\mathbf{R} = \mathbf{R}_{reusable} \cup \mathbf{R}_{consumable}$
- X = Name or unique identifier of a resource, ($X \in \mathbf{R}$)
- $R_{ij}^*(k)$ = Subset of \mathbf{R} with all resources qualified for Requirement j of Task i that is available for the whole duration of the requirement segment if the task starts at slot k and has initial capacity greater or equal than the required number for that task-requirement, i.e., $c(X, k) \geq n_{ij}$.
- s_{ij} = Offset for Requirement j of Task i
- d_{ij} = Number of slots for Requirement j of Task i
- n_{ij} = Number of required resources for Requirement j of Task i
- V_i = Value of Task i
- r_i = Number of Requirements in Task i
- $c(X, k)$ = Capacity of Resource X for its availability interval containing the slot k .
- \mathbf{S}_{T_i} = Set of Starting Intervals for Task T_i . $\mathbf{S}_{T_i} = \mathbf{S}_{T_i}^{(1)}, \mathbf{S}_{T_i}^{(2)}, \dots, \mathbf{S}_{T_i}^{(L(T_i))}$
- $\mathbf{S}_{T_i}^{(l)}$ = Set of possible starting-slots of interval l of Task i

2.1 Variables

In the current version of the encoding we have 4 types of variables:

Name	Description	Number
T_i	<i>Task Activation Variables</i>	N
T_{ik}	<i>Task Starting-Slot Activation Variables</i>	$\sum_i^N \ \mathbf{S}_{T_i}\ $
$X_{i,j}$	<i>Resource Task Requirement Variable</i> Represents the assignment of Resource X to Requirement j of Task i Used both for Reusable and Consumable Resources	
$X_{i,j,k}$	<i>Reusable Resource Slot Variable</i> Represents the assignment of Slot k of the Reusable Resource X to Requirement j of Task i	

2.2 Soft Constraints

$$[\text{soft}] \quad \forall_i T_i \geq 1 \quad \text{for } i = 1, 2, \dots, N. \quad (1)$$

2.3 Task Starting-Slot Selection

The constraint we want to impose is:

$$\bar{T}_i + \sum_{k \in \mathbf{S}_{T_i}} T_{ik} = 1 \quad \text{for } i = 1, 2, \dots, N. \quad (2)$$

Which in the OIP input format translates into the two following inequalities:

$$\bar{T}_i + \sum_{k \in \mathbf{S}_{T_i}} T_{ik} \geq 1 \quad \text{for } i = 1, 2, \dots, N. \quad (3)$$

$$T_i + \sum_{k \in \mathbf{S}_{T_i}} \bar{T}_{ik} \geq \|\mathbf{S}_{T_i}\| \quad \text{for } i = 1, 2, \dots, N. \quad (4)$$

Here \mathbf{S}_{T_i} indicates the set of slots k that are feasible starting slots for the task considering the offsets and length of the segments for all requirements in the task i .

2.4 Resource Selection Constraints

Once a starting slot has been selected by the constraints above, we use the following constraints to pick a resource among the set of possible resources for that starting slot k . At this point we don't distinguish between what instance of resource X we are selecting. We only worry that in its initial availability intervals the resource has enough capacity to accommodate that requirement.

$$\bar{T}_{ik} + \sum_{X \in R_{ij}^*(k)} X_{i,j} \geq 1 \quad \text{for } i = 1, 2, \dots, N \quad (5)$$

$$j = 1, 2, \dots, r_i \quad (6)$$

In this equation we should have $k \in \Sigma(T_i)$ and $X \in R_{ij}^*(k)$ where $\Sigma(T_i)$ is the subset of \mathbf{S}_{T_i} that are feasible starting times for that task. I.e., starting times for which $k \in \mathbf{S}_{T_i}$ and $R_{ij}^*(k) \neq \emptyset \forall j$ in the task. $R_{ij}^*(k)$ is the subset of qualified resources for that requirement with initial availability to accommodate the requirement in terms of capacity and segment duration.

In the previous expression, it is possible to select more than one resource for that requirement of a task. To exclude that possibility we introduce the following additional condition

$$\bar{T}_i + \sum_{X \in \Omega_{ij}} X_{i,j} 1 \text{ for } i = 1, 2, \dots, N \quad (7)$$

$$j = 1, 2, \dots, r_i \quad (8)$$

Where $\Omega_{ij} = \bigcup_{\forall k} R_{ij}^*(k)$

2.5 Reusable Resource Slots Selection Constraints

This constraints are used for *reusable resources only* to turn on d_{ij} consecutive slots (of its availability interval) starting from the $k + s_{ij}$ slot, where s_{ij} is the offset for the requirement's segment

$$d_{ij} \bar{T}_{ik} + d_{ij} \bar{X}_{i,j} + \sum_{l=0}^{d_{ij}-1} X_{i,j,k+s_{ij}+l} \geq d_{ij} \quad (9)$$

$$i = 1, 2, \dots, N$$

$$j = 1, 2, \dots, r_i$$

$$(10)$$

In this equation we should have $k \in \Sigma(T_i)$ and $X \in R_{ij}^*(k)$ where $\Sigma(T_i)$ is the subset of \mathbf{S}_{T_i} that are feasible starting times for that task. I.e., starting times for which $k \in \mathbf{S}_{T_i}$ and $R_{ij}^*(k) \neq \emptyset \forall j$ in the task. $R_{ij}^*(k)$ is the subset of qualified resources for that requirement with initial availability to accommodate the requirement in terms of capacity and segment duration.

This condition alone will in principle enable some spurious slots form the resource availability to be also turned on. To avoid this effect we impose an additional condition to say that we don't want more than d_{ij} slots on for a selected resource and we want all slots turned off if that resource wasn't selected.

$$d_{ij} \bar{X}_{i,j} + \sum_{k \in \mathbf{K}(i,j)} X_{i,j,k} \leq d_{ij} \quad (11)$$

$$\begin{aligned}
i &= 1, 2, \dots, N \\
j &= 1, 2, \dots, r_i
\end{aligned}
\tag{12}$$

Where $\mathbf{K}(i, j)$ is the whole set of feasible slots for the scheduling of resource X to the requirement j of task i , taking into consideration the initial resource availability and capacity of X , the possible task start slots for task i and the segment length, offset and required capacity of requirement j .

2.6 Capacity Exclusion Constraints

In order to have correct solutions we need to preclude the assignment of a resource slot to more requirements than its initial capacity. Since reusable and consumable resources are modelled differently in terms of their slots assignments (i.e., we don't represent consumable resources at the atomic slot level, but only at the range availability level) we have two different expressions for reusable and consumable resources.

$$\sum_{i,j \in X(i,j,k)} n_{ij} X_{i,j,k} \leq c(X, k) \text{ for } X \in \mathbf{R}_{reusable} \tag{13}$$

$$\sum_{i,j \in X(i,j,k)} n_{ij} X_{i,j,k} \leq c(X) \text{ for } X \in \mathbf{R}_{consumable} \tag{14}$$

(15)

where $X(i, j, k)$ is the set of task-requirement pairs in which X is a possible resource for start-slot k .

2.7 Crew Day Constraints

A ZMarbles problem can specify a particular kind of crew day constraints via the following parameters:

- c = crew day length (e.g., 12 hours)
- p = period (e.g., 24 hours)
- s = maximum shift per day (e.g., 2 hours)
- s_{total} = maximum total shift (e.g., 20 hours)
- γ = maximum number of tasks per crew day
- ξ = number of “crew day slots” per slot (positive integer)

In addition, the problem specifies what kind of resources must adhere to crew day rules (typically pilots). The rest of this section describes encoding the crew day constraints for a single resource, a pilot. The same encoding is replicated if there are multiple pilots.

We will also use the following notation:

- ρ = maximum number of crew days that could overlap the planning horizon
- c_l = the start of crew day l ($0 \leq l < \rho$)
- α_l = the smallest legal value of c_l
- β_l = the largest legal value of c_l
- m_l = $\lceil (\alpha_l + \beta_l) / 2 \rceil$
- ρ = maximum number of crew days that could overlap the planning horizon
- $start$ = first slot in the planning horizon
- $last$ = last slot in the planning horizon

All times and durations in this section are measured in “crew day slots,” which are integer multiple of the slots used for resource availability. So if the problem is discretized to 10 minutes for resource availability, crew day slots would be 10ξ minutes.

The crew day constraints can be summarized as follows:

- $-c < -c_0 - start \leq p - c$
- $(\forall l) p - s \leq c_{l+1} - c_l \leq p + s$
- $(\forall l) -s_{\text{total}} \leq c_l - c_0 - lp \leq s_{\text{total}}$
- No work happens outside of a crew day, and no pilot does more than γ tasks in a single crew day.

In order to enforce these constraints, we introduce the following pseudoboolean variables:

$d_{l,b}$	one bit of the difference $c_l - m_l$
A_k	true iff pilot is at rest in slot k
$B_k^{(l)}$	true iff crew day l contains slot k
$Y_{i,k}$	true iff pilot does task i in slot k
$T_{i,l}$	true if (not iff) this pilot does task i on crew day l

The quantity c_l is not a pseudoboolean variable, but it may be expressed as

$$m_l - 2^n d_{l,n} + d_{l,0} + 2d_{l,1} + \dots + 2^{n-1} d_{l,n-1} \quad ,$$

where the value of n is determined by how large a range must be representable (i.e., by $\beta_l - \alpha_l$).

Given that, we can trivially encode

$$\begin{aligned} -c &< -c_0 - \text{start} \leq p - c, \\ p - s &\leq c_{l+1} - c_l \leq p + s, \\ -s_{\text{total}} &\leq c_l - c_0 - lp \leq s_{\text{total}}, \text{ and} \\ (c_l > \text{last}) &\rightarrow (c_l - c_{l-1} = p). \end{aligned}$$

The last of those is merely to reduce the number of redundant, equivalent solutions.

For the A 's and B 's we encode

$$\begin{aligned} A_k + B_k^{(0)} + \dots + B_k^{(\rho-1)} &= 1 \text{ and} \\ B_k^{(l)} &= (c_l \leq k < c_l + c). \end{aligned}$$

(The latter expands into a few pseudoboolean constraints.) These constraints ensure that each slot is assigned to exactly one crew day or to “rest.”

If crew day slot k corresponds to resource slots x through $x + \xi - 1$ then we encode

$$\begin{aligned} Y_{i,k} &\leftrightarrow (\sum X_{i,j,x} + \sum X_{i,j,x+1} + \dots + \sum X_{i,j,x+\xi-1} > 0) \text{ and} \\ Y_{i,k} &\rightarrow \overline{A_k} \end{aligned}$$

to ensure that no work occurs when A_k is true.

Finally, to limit the number of tasks per crew day, we encode

$$T_{0,l} + T_{1,l} + \dots \leq \gamma$$

and

$$B_k^{(l)} \wedge Y_{i,k} \rightarrow T_{i,l} \quad .$$

3 Pseudo-Boolean Original Encoding: Version with Capacity Instances

This section describes the original PB encoding of the version that was demo at the ANT's PI meeting.

- N = Number of Tasks
- M = Number of Resources
- \mathbf{R} = Set of all resources ($X \in \mathbf{R}$)
- $R_{ij}(k)$ = Subset of \mathbf{R} with all resources qualified for Requirement j of Task i that is available for the whole duration of the requirement segment if the task starts at slot k .
- \mathbf{R}_{RU} = Set of Reusable Resources
- \mathbf{R}_C = Set of Consumable Resources
- s_{ij} = Offset for Requirement j of Task i
- d_{ij} = Number of slots for Requirement j of Task i
- n_{ij} = Number of required resources for Requirement j of Task i
- V_i = Value of Task i
- r_i = Number of Requirements in Task i
- $c_l(X)$ = Capacity of the Availability Interval l of Resource X
- \mathbf{S}_{T_i} = Set of Starting Intervals for Task T_i . $\mathbf{S}_{T_i} = \mathbf{S}_{T_i}^{(1)}, \mathbf{S}_{T_i}^{(2)}, \dots, \mathbf{S}_{T_i}^{(L(T_i))}$
- $\mathbf{S}_{T_i}^{(l)}$ = Set of possible starting-slots of interval l of Task i
- $L(X)$ = Number of availability range slots for resource X

3.1 Variables

In the current version of the encoding we have 3 types of variables:

Name	Description	Number
T_i	<i>Task Activation Variables</i>	N
T_{ik}	<i>Task Starting-Slot Activation Variables</i>	$\sum_i^N \ \mathbf{S}_{T_i}\ $
$X_{i,j,k}^{(m)}$	<i>Resource Instance Slot Variable</i> Represents the assignment to Slot k of the instance m of Resource X to Requirement j of Task i	

3.2 Soft Constraints

$$[soft] \quad V_i T_i \geq 1 \quad for \quad i = 1, 2, \dots, N. \quad (16)$$

3.3 Task Starting-Slot Selection

The constraint we want to impose is:

$$\bar{T}_i + \sum_{k \in \mathbf{S}_{T_i}} T_{ik} = 1 \quad for \quad i = 1, 2, \dots, N. \quad (17)$$

Which in the OIP input format translates into the two following inequalities:

$$\bar{T}_i + \sum_{k \in \mathbf{S}_{T_i}} T_{ik} \geq 1 \quad for \quad i = 1, 2, \dots, N. \quad (18)$$

$$T_i + \sum_{k \in \mathbf{S}_{T_i}} \bar{T}_{ik} \geq \|\mathbf{S}_{T_i}\| \quad for \quad i = 1, 2, \dots, N. \quad (19)$$

Here \mathbf{S}_{T_i} indicates the set of slots k that are feasible starting slots for the task considering the offsets and length of the segments for all requirements in the task i .

3.4 Requirement-First-Slot Selection

Once a starting slot has been selected by the constraints above, we use the following constraints to pick the first slot among the possible set of resources. Since for each requirement we need a required number n_{ij} of resources we have to try to get at least that number of instances from a single resource.

The first constraint we want to impose is:

$$n_{ij} \bar{T}_{ik} + \sum_{X \in R_{ij}(k)} \sum_{m=1}^{c_l(X)} X_{i,j,k+s_{ij}}^{(m)} \geq n_{ij} \quad for \quad i = 1, 2, \dots, N \quad (20)$$

$$j = 1, 2, \dots, r_i$$

$$k \in \mathbf{I}^{(l)}(X, i, j)$$

$$l = 1, 2, \dots, L(X)$$

Now, this constraint can be satisfied by picking n_{ij} variables corresponding to instances from different resources. To force all variables to come from the same resource, we add the following two constraints:

$$(n_{ij} - 1) \bar{T}_{ik} + (n_{ij} - 1) \bar{X}_{i,j,k+s_{ij}}^{(m)} + \sum_{l=1(l \neq m)}^{c_l(X)} X_{i,j,k+s_{ij}}^{(l)} \geq (n_{ij} - 1) \quad for \quad i = 1, 2, \dots, N \quad (21)$$

$$j = 1, 2, \dots, r_i$$

$$\begin{aligned}
k &\in \mathbf{I}^{(l)}(X, i, j) \\
l &= 1, 2, \dots, L(X)
\end{aligned}$$

and

$$\begin{aligned}
(\mathcal{S} - n_{ij})\bar{T}_{ik} + \sum_{X \in R_{ij}(k)} \sum_{m=1}^{c_i(X)} \bar{X}_{i,j,k+s_{ij}}^{(m)} &\geq (\mathcal{S} - n_{ij}) \text{ for } i = 1, 2, \dots, N \\
j &= 1, 2, \dots, r_i \\
k &\in \mathbf{I}^{(l)}(X, i, j) \\
l &= 1, 2, \dots, L(X)
\end{aligned}$$

where \mathcal{S} is the total number of terms in the double sum in the second term of the lhs.

3.5 Requirement Segment Slots Selection

For Reusable Resources we want to turn on d_{ij} consecutive slots (of its availability interval) after the starting slot. We use the following constraints:

$$\begin{aligned}
(d_{ij} - 1)\bar{T}_{ik} + (d_{ij} - 1)\bar{X}_{i,j,k+s_{ij}}^{(m)} + \sum_{l=1}^{d_{ij}-1} X_{i,j,k+s_{ij}+l}^{(m)} &\geq (d_{ij} - 1) \quad (22) \\
i &= 1, 2, \dots, N \\
j &= 1, 2, \dots, r_i \\
k &\in \mathbf{S}_X \\
m &= 1, 2, \dots, c_X
\end{aligned}$$

For Consumable Resources we want to block or turn ON all slots in the availability interval corresponding to the starting slot k so the constraint should be modified to:

$$\begin{aligned}
(\Omega(X, k) - 1)\bar{T}_{ik} + (\Omega(X, k) - 1)\bar{X}_{i,j,k+s_{ij}}^{(m)} + \sum_{l=f}^{f+\Omega(X,k)-1} X_{i,j,l}^{(m)} &\geq (\Omega(X, k) - 1) \quad (23) \\
i &= 1, 2, \dots, N \\
j &= 1, 2, \dots, r_i \\
k &\in \mathbf{S}_X ; m = 1, 2, \dots, c_X
\end{aligned}$$

Where $\Omega(X, k)$ is the number of slots in the availability range of resource X for starting slot k .

3.6 Exclusion Constraints

In order to have correct solutions we need to preclude assigning a the instance of a resource slot to more than one Task-Requirement at a time. We encode that with the following constraints:

$$\sum_{i,j \in X(i,j,k)} X_{i,j,k}^{(m)} \leq 1 \text{ for } X \in \mathbf{R} \quad (24)$$
$$m = 1, 2, \dots, c_l(X)$$

where $X(i, j, k)$ is the set of task-requirement pairs in which X is a possible resource for start-slot k .

4 Appendix: Pseudo-Boolean algebra

PB variables take values $x = 0$ or 1 and there is a relation between the variable and its negation as follows:

$$\bar{x} = 1 - x \quad (25)$$

where \bar{x} is the negation of x .

Most constraints in the encoding of Z-marbles problem are of equality form as follows:

$$(a)\bar{X} + (b) \sum_{i=1}^N Z_i = c \quad (26)$$

where a, b and c are numbers and X and Z_i are PB variables.

By the input format requirement of the solver, constraints must be the inequalities of the form $lhs \geq rhs$. We can encode the problem into the right format by translating each equality constraint at first into the following two inequality constraints:

$$(a)\bar{X} + (b) \sum_{i=1}^N Z_i \geq c \quad (27)$$

$$(a)\bar{X} + (b) \sum_{i=1}^N Z_i \leq c \quad (28)$$

Second, because the last expression is not in the required input format and therefore it must be converted into the right form by introducing the negation of corresponding variables as follows:

$$(a)X + (b) \sum_{i=1}^N \bar{Z}_i \geq (bN + a - c) \quad (29)$$