# An innovative method to solve the maintenance task allocation & packing problem - Algorithms

J.N.M. Filho, A.C.P. Mesquita, G.C. Rocha, F.T.M. Abrahão

*Instituto Tecnológico de Aeronáutica - DCTA/ITA/IEMC*
*Praçça Mal. Eduardo Gomes, 50*
*São José dos Campos - SP - 12.228-900 - Brazil*

## 1. Introduction to Algorithms

This annex provides further information regarding the algorithms that were utilized in the research article titled *An innovative method to solve the maintenance task allocation & packing problem*.

## 2. Algorithms

We created a new method which we named ETAPPS (Efficient TAPP Solver) to deal with the TAPP in an efficient and fast way, which is described by the algorithms that follows.

Algorithm 1 initialises constants, loads sets from files and creates the set of work packages. All capitalized constants are considered of global scope, being available for the other algorithms without the need of passing them as arguments.

*Email addresses:* nogueira@ita.br (J.N.M. Filho), celio@ita.br (A.C.P. Mesquita), grocha@ita.br (G.C. Rocha), abrahao@ita.br (F.T.M. Abrahão)

**Algorithm 1** *Initialize()*

```
1: $Y \leftarrow 5$                                    ▷ 5, 10, 15 years or more
2: $Prof \leftarrow 1$                                 ▷ 1, 2 or 3, depending on the operational level
3: $OoP \leftarrow \{750,\ 1500,\ 2500\}$              ▷ flight hours per year
4: $H \leftarrow OoP[Prof] \times Y$                   ▷ Calculates the flight hour horizon
5: $OoPfactor \leftarrow \{1.0,\ 1.4,\ 1.8\}$          ▷ Out-of-Phase cost factor
6: $withR \leftarrow \{True,\ False\}$                 ▷ If True considers the component reliability on optimization
7: $DOC \leftarrow 70,000.000$                         ▷ Daily Opportunity Cost
8: $OHD \leftarrow \frac{H}{Y \times 365}$             ▷ Operating Hours per Day
9: $HOC \leftarrow \lfloor \frac{DOC}{OHD} \rfloor$    ▷ hourly Opportunity Cost
10: $CMCF \leftarrow 3.0$                              ▷ Corrective Maintenance Cost Factor
11: $CMTF \leftarrow 1.2$                              ▷ Corrective Maintenance Time Factor
12: $C,\ M,\ Z,\ P \leftarrow$ load sets from files
13: $STEP \leftarrow \min_{t=1}^{|C|}(lim_t)$
14: $OoPStep \leftarrow \frac{STEP}{10}$
15: $numStops \leftarrow \lceil \frac{H}{STEP} \rceil$
16: $S \leftarrow \{\}$                                ▷ set of regular maintenance stoppages
17: $OoP \leftarrow \{\}$                              ▷ set of out-of-phase maintenance stoppages
18: $numOoPsteps = \lceil \frac{STEP}{OHD} \rceil$     ▷ number of OoP steps in days
19: for $i \leftarrow 1$ to $numStops$
20:     $stop_i \leftarrow i \times STEP$
21:     $B^i \leftarrow \{\}$
22:     $stoppage \leftarrow (stop_i,\ dt_i \leftarrow 0,\ cost_i \leftarrow 0,\ B^i)$
23:     $S \leftarrow S \cup \{stoppage\}$
24:     for $p \leftarrow 1$ to $numOoPsteps - 1$      ▷ for each regular stoppage create a number of OoP stoppages
25:         $OoP \leftarrow OoP \cup \{stop_i + OHD * p\}$
26: return $C,\ M,\ Z,\ P,\ S,\ OP$
```

Line 7 initializes Daily Opportunity Cost constant $DOC$ and line 8 the number of operating hours per day that may receive values according to the client aircraft exhaustion level selected. Line 9 initializes Hourly Opportunity Cost constant $HOC$.

The $CMCF$ (line 10) is a corrective factor applied to the preventive maintenance cost to predict the corrective cost in case some repair is made necessary during preventive maintenance. The same applies to the $CMTF$ to predict the corrective maintenance duration.

Line 12 loads data into components $C$, preparations $P$, and qualifications $M$ and line 4 defines the packing horizon as a function of the maximum flight hour limit among components. Line 15 calculates the number of maintenance stoppages in this horizon. Lines 16 to 23 creates the set of maintenance work packages.

All constants referred to in this algorithm may be updated through a maintenance and failures history analysis made by appropriate machine learning tools, according to the dynamics of operations and scenario changing.

### *2.1. The main algorithm*

In Algorithm 2, we define 2 resolution methods: (1) a simple heuristic (*Simple*) to emulate engineers' steps in manually solving the TAPP. This heuristic allocates tasks to work packages with the only concern being keeping components from flying after their due flight hour; and (2) we used a MIP solver and a *First-Fit Decreasing* (FFD) approximation algorithm that handles the same issues as the engineers, but with an efficient account of resources per work package, where resources were not accounted more than once per work package; checks if the number and qualifications available attend tasks needs; and also, in the FFD phase, that incompatible and precedent tasks are not executed in the same *Bin* attempting also to optimize availability.

**Algorithm 2** *Main()*

```
1:  $C,\ M,\ Z,\ P,\ S \leftarrow Initialize()$
2:  $methods \leftarrow \{Simple, ETAPPS\}$
3:  $useFFD \leftarrow \{False, True\}$
4:  $iter \leftarrow 20$
5:  $MC_{tot} \leftarrow 0$
6:  $DT_{tot} \leftarrow 0$
7:  for $method \in methods$
8:      for $ffd \in useFFD$
9:          for $it \leftarrow 1$ to $iter$
10:             $T \leftarrow CreateTasks(C,\ M,\ Z,\ P)$
11:             $MC,\ DT \leftarrow Solve(method,\ S,\ T,\ ffd,\ C)$
12:             $MC_{tot} \leftarrow MC_{tot} + MC$
13:             $DT_{tot} \leftarrow DT_{tot} + DT$
14:         $MC \leftarrow \frac{MC_{tot}}{iter}$                ▷ calculate the average cost
15:         $DT \leftarrow \frac{DT_{tot}}{iter}$
16:         $A \leftarrow \frac{H-DT}{H}$                ▷ calculate the availability
17:         print $method,\ ffd,\ MC,\ A$
```

It is important to emphasize that all constants referred to (initialized by Algorithm 1) are considered global scope. This is why they are not passed as function arguments. An exception occurs when some argument is changed locally. e.g., $T$ and $C$ in line 10.

Line 1 initializes all constants and sets referred to in this Algorithm (2), according to Algorithm 1.

To simulate cycles of plan, execute, and record maintenance, we defined a number of iterations (line 4), that creates tasks and solves the TAPP as a means of exploring the emulated real world events. Each cycle is composed of many maintenance events (or work packages) sequentially executed in a time horizon.

Line 9 solves *iter* times with the same method and accumulates the maintenance costs $MC_{tot}$ and the downtime $DT_{tot}$, to be divided by *iter* lately to calculate the averages.

Line 11 solves the TAPP with one of the methods, returning the maintenance costs $MC$ and the downtime $DT$.

Line 17 for each method, prints the estimated total maintenance cost and availability for the planned horizon.

## *2.2. The tasks creation algorithm*

In Algorithm 3 we use 3 types of random selections: (1) *RandomReal* which selects in a continuous range of real numbers, (2) *RandomInteger* that selects in a discrete range of integers, and (3) $RandomChoice(set,\ number)$ that selects elements from a set, where the second parameter is the number of elements to be chosen. Algorithm 3 creates a set of tasks based on the sets of preparations, components and qualifications.

**Algorithm 3** *CreateTasks(C, M, Z, P)*

```
1:  $T \leftarrow \{\}$
2:  for $j \leftarrow 1$ to $|C|$
3:      $c \leftarrow C[j]$                                              ▷ c is a component
4:      $c.last \leftarrow 0$                    ▷ reset the last component stoppage (flight hour)
5:      $mat_j \leftarrow c.mat \times (1.0 + RandomReal[-0.2, 0.2])$
6:      $mh_j \leftarrow mh_t \times (1.0 + RandomReal[-0.2, 0.2])$
7:      $qualif_j \leftarrow RandomInteger[1,\ |M|]$
8:      $nmec_j^r \leftarrow RandomInteger[1,\ M[qualif_j].available]nmec_j^r$
9:      $pmdt_j \leftarrow \frac{mh_j}{\sum_{r=1}^{|M|} mh_j^r}$
10:     $cmdt_j \leftarrow \frac{mh_j \times CMTF}{\sum_{r=1}^{|M|} mh_j^r}$
11:     $pmc_j \leftarrow \sum_{r=1}^{|M|} mh_j^r \times wage^r + HOC \times pmdt_j + mat_j$
12:     $cmc_j \leftarrow \sum_{r=1}^{|M|} mh_j^r \times CMTF \times wage^r + cmdt_j \times HOC$
13:     $R_t^i \leftarrow e^{-(\frac{stop_i}{\eta_t})^{\beta_t}}$ probability of success (no findings related do item $c_t$) on the preventive maintenance
14:     $R_t^p \leftarrow e^{-(\frac{stop_p}{\eta_t})^{\beta_t}}$ probability of success (no findings related do item $c_t$) on the out of phase maintenance
15:     $pmc_j^i \leftarrow (R_t^i) \times pmc_j$ PM cost in package $s_i$
16:     $pmdt_j^i \leftarrow (R_t^i) \times pmdt_j$ PM downtime cost in package $s_i$
17:     $cmc_j^i \leftarrow (1 - R_t^i) \times cmc_j$ CM cost in package $s_i$
18:     $cmdt_j^i \leftarrow (1 - R_t^i) \times cmdt_j$ CM downtime cost in package $s_i$
19:     $pmc_j^p \leftarrow (R_t^p) \times pmc_j$ PM cost in OoP package $o_p$
20:     $pmdt_j^p \leftarrow (R_t^p) \times pmdt_j$ PM downtime cost in OoP package $o_p$
21:     $cmc_j^p \leftarrow (1 - R_t^p) \times cmc_j$ CM cost in OoP package $o_p$
22:     $cmdt_j^p \leftarrow (1 - R_t^p) \times cmdt_j$ CM downtime cost in OoP package $o_p$
23:     $incompatible_j \leftarrow -1$                                   ▷ no incompatible task
24:     $startAfter_j \leftarrow -1$                                       ▷ no precedent task
25:     if $j \mod 30 = 0$                                          ▷ if the remainder is zero
26:         $incompatible_j \leftarrow RandomInteger[j-30,\ j-1]$    ▷ 1 of the last 29 tasks is incompatible with $j$
27:         $startAfter_j \leftarrow RandomInteger[j-30,\ j-1]$  ▷ $j$ must start after 1 of the last 29 tasks is finished
28:     $zone_j \leftarrow RandomInteger[1,\ |Z|]$
29:     $t_j \leftarrow (pmc_j, pmdt_j, cmc_j, cmdt_j, zone_j, qualif_j, nmec_j, preps_j, startAfter_j, incompatible_j)$
30:     $T \leftarrow T \cup \{t_j\}$
31: return $T$
```

Each task must not be executed concurrently with its incompatible or precedent tasks (if it has some). So, lines 25 to 27 randomly selects the incompatible and precedent tasks.

The last maintenance stoppage of this component is initialized in line 4. This value is updated along TAPP resolution to simulate a series of maintenance stoppages in a flight time horizon.

Finally, in line 30 this task vector is added to the set under assembly which is returned in line 31. Set $C$ is also returned because parameter $last_t$ has been initialized.

## *2.3. The Simple method for TAPP*

**Algorithm 4** *Simple(S, T, C)*

```
$X_{ij} \leftarrow 0$, for $i \in \{1, 2, ..., |S|\}$, for $j \in \{1, 2, ..., |T|\}$
for $j \leftarrow 1$ to $|T|$
    $t \leftarrow cid_j$
    for $i \leftarrow 1$ to $|S|$
        $flyUntil = C[t].last + C[t].lim$
        if $stop_i <= flyUntil$ and $flyUntil < stop_i + STEP$
            $X_{ij} \leftarrow 1$
            $C[t].last \leftarrow i$
Return $X_{ij}$
```

## *2.4. The First-Fit Decreasing algorithm*

Algorithm 5 places a special implementation of the *First-Fit Decreasing (FFD)* solution method for the *Bin Packing Problem*. It minimizes the number of tasks bins, minimizing the work package downtime.

**?** investigates the *First-Fit Decreasing (FFD)* algorithm in this doctoral thesis, with the main result being a proof that the *FFD* for the bin packing problem never returns a solution that uses no more than $(11/9 \times OPT)$ bins, where $OPT$ is the optimal number of bins. Later, **?** propose a new version that returns a solution that uses no more than $(71/60 \times OPT)$ bins.

**Algorithm 5** $FFD(i,\ X_{ij},\ T,\ M,\ Z)$

```
$reserved \leftarrow 0$
$bins \leftarrow \{\}$
$b \leftarrow 1$
$Bin_b \leftarrow \{\}$
$bins \leftarrow bins \cup \{Bin_b\}$
$W \leftarrow [0]$                                        ▷ a matrix with $|T|$ rows and $|T|$ columns
sort$_{j=1}^{|T|}(nmec_j + Z[zone_j].limit,\ decreasing)$
for $j \leftarrow 1$ to $|T|$
    if $X_{ij} = 1$
        $NotIncluded \leftarrow$ True
        for $Bin_b \in bins$
            $needed \leftarrow reserved + nmec_j$
            if $W_{jb} = 0$ and $needed \leq M[qualif_j].available$ and $needed \leq Z[zone_j].limit$
                if $W_{incompatible_j b} = 0$ and $j > startAfter_j$
                    $Bin_b \leftarrow Bin_b \cup \{t_j\}$
                    $reserved \leftarrow reserved + nmec_j$
                    $NotIncluded \leftarrow$ False
                    $W_{jb} \leftarrow 1$
                    break
        if $NotIncluded$
            $reserved \leftarrow 0$
            $b \leftarrow b + 1$
            $Bin_b \leftarrow \{\}$
            $Bin_b \leftarrow Bin_b \cup \{t_j\}$
            $bins \leftarrow bins \cup \{Bin_b\}$
            $W_{jb} \leftarrow 1$
return $bins$
```

Some remarks on Algorithm 5:

Line 7 sorts tasks by the decreasing order of mechanics need.

Lines 1 and 21 initialize the number of reserved mechanics of qualification $r$ for the task, as it is the size a *Bin* of tasks.

In line 2 a bin for each mechanic qualification is created. This is necessary because each qualification has its available number of mechanics, that will be the size of each bin. Until line 5 $|M|$ sets with 1 empty bin each are initialized.

In line 9, it is checked if the task $j$ is associated to the package $i$ to try task inclusions in any bin.

From line 11 until 19 the set of existent bins is iterated in a try to include a task. From line **??** until **??**, if the task inclusion is feasible, it is included in a bin and variable $W_{jb}$ is set to indicate this inclusion. Also variable $reserved^r$ is updated for feasibility check on later inclusion tries.

From line 20 to 25, if no task is included, a new empty bin is created and inserted in the set of bins.

We conducted some preliminary tests by trying to fit 50 to 200 items in as few bins as possible, indicating that FFD was likely to obtain solutions optimal or very close for most of the tests.

**Algorithm 6** *Solve(method, S, T)*

```
1:  $MC, DT \leftarrow 0, 0$
2:  $X_{ij} \leftarrow 0$, for $i \in \{1, 2, ..., |S|\}$, for $j \in \{1, 2, ..., |T|\}$
3:  $O_{pj} \leftarrow 0$, for $p \in \{1, 2, ..., |O|\}$, for $j \in \{1, 2, ..., |T|\}$
4:  $W_{jb} \leftarrow 0$, for $j \in \{1, 2, ..., |T|\}$, for $b \in \{1, 2, ..., |B^i|\}$, for $i \in \{1, 2, ..., |S|\}$
5:  if $method = ETAPPS$
6:      $X_{ij},\ O_{pj} \leftarrow Branch\&Cut.minimize()$
7:  if $method = Simple$
8:      $X_{ij} \leftarrow SimpleSolve(S,\ T)$
9:  if $ffd = True$
10:     for $i \leftarrow 1$ to $|S|$
11:         $B^i \leftarrow FFD(i,\ X_{ij},\ T,\ M,\ Z)$
12:         for $bin \in B^i$
13:             $dt_{bin} \leftarrow 0$
14:             for $t_j \in bin$
15:                 if $pmdt_j + cmdt_j > dt_{bin}$
16:                     $dt_{bin} \leftarrow pmdt_j + cmdt_j$
17:             $DT \leftarrow DT + dt_{bin}$
18: else
19:     $DT \leftarrow DT + pmdt_j + cmdt_j$
        State $preps_i \leftarrow \{\ \}$                    ▷ set of unique preparations for package $i$
20: for $i \leftarrow 1$ to $|S|$
21:     for $j \leftarrow 1$ to $|T|$
22:         if $X_{ij} = 1$
23:             $last_t \leftarrow stop_i$
24:             $MC \leftarrow MC + pmc_j + cmc_j$
25:         else if $O_{pj} = 1$
26:             $last_t \leftarrow stop_i$
27:             $MC \leftarrow MC + pmc_j + cmc_j$
28:     for $prep \in preps_j$
29:         if $prep \notin preps_i$                   ▷ guarantee that no preparation is duplicated
30:             $preps_i \leftarrow preps_i \cup prep$
31:             $MC \leftarrow MC + prep.cost$
32:             $DT \leftarrow DT + prep.dt$
33: $A \leftarrow \frac{H-DT}{H}$
34: Return $MC,\ DT$
```

Some remarks on Algorithm 6: In line 1, the variables for maintenance cost ($MC$) and downtime ($DT$) are initialized. Lines 2-4 initialize the decision variables. In line 6, the *CoIn-Or Branch & Cut* solves TAPP and returns variables $X_{ij}$ and $O_{pj}$ set. In line 11, the number of bins is minimized by the *First Fit Decrease (FFD)* (**?**) method and returns variables $W_{jb}$ set. In line 8, the *Simple* method solves TAPP and returns variables $X_{ij}$ set. Lines 20-32 add subtask costs to the maintenance cost and totalize downtime. In line 33, the availability ($A$) is calculated. Line 34 returns the optimized maintenance cost and calculated availability.