**EPC Linear Model**.

Let us recall the input data for the **EPC** problem:

|  |
| --- |
| **Vehicle related input** |
| *M*:number of stations (*Depot* excluded) |
| *Depot****Depot = M* + 1): vehicle tour (without refueling) |
| *TMax*: maximal time for the vehicle to achieve its tour |
| *CVeh*:vehicle tank capacity |
| *E0*: initial vehicle hydrogen load  For *j* = 0,…, *M, tj*: required time to go from station *j* to station *j* + 1  For *j* = 0,…, *M,* *dj*: required time to go from station *j* to the micro-plant |
| For *j* = 0,…, *M, d\*j*: required time to go from the micro-plant to station *j*  For *j* = 0,…, *M, ej*: required energy to go from station *j* to station *j* + 1  For *j* = 0,…, *M,* *εj*: required energy to go from station *j* to the micro-plant  For *j* = 0,…, *M, ε\*j*: required energy to go from the micro-plant to station *j* |
| **Micro-plant production related input** |
| *CMP*: micro-plant tank capacity |
| *N*:number of production periods |
| *p:* duration (in time units) of one production period |
| *H0*: initial micro-plant hydrogen load |
| *CostF*: activation cost |
| For *i* = 0,…, *N* –1*, Pi =* [*p*.*i*, *p*.(*i* + 1)[: time interval related to production period *i* |
| For *i* = 0,…, *N* –1*, Ri*: production rate related to period *i* |
| For *i* = 0,…, *N* –1*, CostVi*: production cost related to period *i* |

Table 1. *Input data for the EPC problem*

1. **An Integrated Mathematical Programming (MP) Model**

*MP* is not well-fitted to **EPC**. Still, we may use it in order to formulate our problem in an unambiguous way, based upon 3 main variables:

* **Production variables**:
  + *z* = (*zi*, *i* = –1,…, *N* – 1), with {0, 1} values: *zi*= 1 ~ the micro-plant is active during period *i* (*i* = –1 corresponds to a fictitious period);
  + *y* = (*yi*, *i* = 0,…, *N* – 1), with {0, 1} values: *yi* = 1 ~ the micro-plant is activated at the beginning of *i*;
  + *VTan*k = (*VTanki*, *i* = 0,…, *N*), with non negative integer values: *VTanki*is the hydrogen load of the *micro-plant* tank at the beginning of period *i*; We involve here a fictitious period *N* in order to express the fact that the load of micro-plant tank at the end of the process should be at least equal to *H*0;
  +  = (*i*, *i* = 0,…, *N* – 1), with {0, 1} values: *i* = 1 ~ the vehicle is refueling during period *i*;
  + *L\** = (*L\*i*, *i* = 0,…, *N* – 1), with non negative integer values: in case *i* = 1, *L\*i* is the quantity of hydrogen loaded by the vehicle during period *i*; else, *L\*i* may take any non negative value.
* **Vehicle variables**:
  + *x* = (*xj*, *j* = 0,…, *M*), with {0, 1} values: *xj* = 1 ~ the vehicle refuels while traveling from station *j* to station *j* + 1;
  + *L* = (*Lj*, *j* = 0,…, *M*), with non negative integer values: if *xj* = 1, *Lj* = hydrogen quantity loaded by the vehicle while traveling from *j* to *j* + 1; else it may take any non negative value;
  + *T* = (*Tj*, *j* = 0,…, *M* + 1), with non negative integer values: *Tj* = time when the vehicle arrives at *j*;
  + *T\** = (*T\*j*, *j* = 0,…, *M* + 1), with non negative integer values: if *xj* = 1, *T\*j* = time when the vehicle starts refueling while traveling from *j* to *j* + 1; else it may take any non negative value;
  + *VVeh* = (*VVehj*, *j* = 0,…, *M* + 1), with non negative integer values: *VVehj* = hydrogen load of the vehicle tank when the vehicle arrives in *j*.
* **Synchronization variables**: *U* = (*Ui,j,* *i* = 0,…, *N* – 1, *j* = 0,..., *M*) with {0, 1} values: *Ui,j* = 1 ~ the vehicle is going to refuel during period *i* while traveling from *j* to *j* + 1.

Constraints come as follows (for a better understanding, we use here a logical formulation, easy to linearize through *Big M* technique):

* **Objective function**: Minimize

 *i* = 0,...,*N* – 1 (*CostF.yi* + *CostVi..zi*) + .*TM* + 1 .

* **Production constraints**:
  + For any *i* = 0,…, *N* – 1: *yi* = 1 → (*zi* = 1 ∧ *zi –* 1 = 0);
  + For any *i* = 0,…, *N* – 1: *zi* + *i* ≤ 1;
  + z –1 = 0;
  + *VTank*0 = *H*0; *VTankN* ≥ *H*0;
  + For any *i* = 0,…, *N* – 1: *VTanki* ≤ *CMP* ;
  + For any *i* = 0,…, *N* – 1: *VTanki* + 1 = *VTanki* + *zi.Ri* – *i.L\*i*.
* **Vehicle Constraints**:
  + *T*0 = 0; *VVeh*0 = *E*0; *VVehM* + 1 ≥ *E*0;
  + For any *j* = 1,…, *M* + 1: *VVehj* ≤ *CVeh*;
  + For any *j* = 0,…, *M*: *VVehj* ≥ *j*; (E1)

(E1) means that at any time, the vehicle must be able to go to the *micro-plant* and refuel, and relies on the *Triangle Inequality* for energy coefficients *ej* and *j*;

* + For any *i* = 0,…, *M*: *Lj* ≤ *CVeh* + *j* – *VVehj*; (E2)

(E2) expresses the fact that the vehicle cannot refuel more than the space which remains inside its tank;

* + For any *j* = 0,…, *M*: *Tj* + 1 ≥ (1 – *xj*).(*Tj*+*tj*)+ *xj*.(*T\*j* + *p* + *d\*j* + 1); (E3)
  + For any *j* = 0,…, *M*: *T*\**j*  ≥  *Tj*+*dj*; (E4)
  + For any *j* = 0,…, *M*: *xj* = 0 → *VVehj* + 1 = *VVehj* – *ej*;
  + For any *j* = 0,…, *M*: *xj* = 1 → *VVehj* + 1 = *VVehj* – *j* – *j* + 1 + *Lj*;
  + *TM* + 1 ≤ *TMax*.
* **Synchronization constraints**:
  + For any *j* = 0,…, *M*: *i* = 0,..., *N* – 1 *Ui,j* = *xj*; (E5)
  + For any *i* = 0,…, *N* – 1, *i* = *j* = 0, ..., *M* *Ui,j*; (E6)
  + For any *j* = 0,…, *M*, *xj* = 1 → *T\*j*= *i*= 0, ..., *N* – 1 *p.i.Ui,j*; (E7)
  + For any *i* = 0,…, *N* – 1: *L\*i* ≤ *VTanki* (E8)

(E8) expresses that load *L\*i* cannot exceed the current load of the micro-plant tank;

* + For any *j* = 0, …, *M*: *Lj* =  *i*= 0,..., *N* – 1 *Ui,j.L\*i.* (E9)

We may state:

**Theorem 1**: *Solving above* ***MP\_EPC*** *model also solves our* ***EPC*** *problem*.

**Proof**: Checking that a feasible solution (*y*, x, *T*, *L*) of **EPC** can be turned into a feasible solution of above linear model with the same cost comes in a straightforward way. We only need to follow the trajectory induced by (*y*, , x, *L*) and compute *z*, *L*\*, *T*, *T*\*, *VTank*, *VTank*, *U*, accordingly.

Conversely, let us consider some feasible solution (*y*, , x, *T*, *L*, *z*, *L*\*, *T*\*, *VTank*, *VTank*, *U*) of above linear model. The key point is that vector *U* defines a matching *i* -> *j*(*i*) between *I*° = {*i* ∈ 0,…, *N*-1, such that *i* = 1} and *J*° = {*j* ∈ 0,…, *M*, such that *xj* = 1} and that this matching is consistent with standard linear ordering: if *i*1 < *i*2 then *j*(*i*1) < *j*(*i*2). The first point is contained into equations (E5, E6).The second point derives from equations (E7), which fixes values *T\*j*(*i*)  and inegalities (E3, E4): if *i*1, *i*2 are consecutive in *I*° and such that *j*(*i*1) > *j*(*i*2), then we get, by propagating (E3, E4), *T\*j*1 ≥ *T\*j*2 and a contradiction with (E7).

It comes that, if (*y*, , x, *T*, *L*, *z*, *L*\*, *T*\*, *VTank*, *VTank*, *U*) is optimal, we see that (E3, E4) are going to give rise to equalities, which means that *T* and *T*\* are going to follow the EPC trajectory induced by (*y*, , x, *L*). But we also see that related load *Lj*(*i*) = *L\*i* (because of (E9)) are going to be feasible in the sense that they should exceed neither the load of the micro-plant tank at the beginning of period *i*, nor the difference between the capacity of vehicle tank and its current load when its arrive to the micro-plant, while moving from *j* to *j*+1, because (E2) and (E8)). We conclude that our solution (*y*, , x, *T*, *L*, *z*, *L*\*, *T*\*, *VTank*, *VTank*, *U*) may be interpreted as an **EPC** trajectory, with the same value. €

Let us pay attention now to the linearization **Linear-EPC**, through *Big M* techniques, of above **MP\_EPC** model, and its rational relaxation. Let us suppose that we reformulate any implications:

* *X* = 0 -> *Y* ≥ 0
* *X* = 1 -> *Y* ≤ 0

as:

* *X* + *Y*/*Big\_M* ≥ 0
* *X* + *Y*/*Big*\_*M* ≤ 1

where *Big*\_*M* is a very large number.

Then we see that:

**Proposition 1**: *According to this hypothesis, the optimal value of the rational relaxation of Linear-EPC is null*.

**Proof**: It is enough to check that, if *Big\_M* is choosen large enough, then we get a feasible solution of the rational relaxation of **Linear\_EPC** by setting:

* *xj* = ½ for every *j*; *i* = (*M*+1)/2N for any *i*;
* *Ui,j* = 1/2*N* for any *i*, *j*;
* *zi* = *yi* = 0 for any *i*;
* *Lj* = *L\*i* = 0 for any *i*;
* *VTanki**H*0 for any *i*;*VVehj**E*0 for any *j*;
* *Lj* = 0 and *T*\**j* = *dj* for any *j*.

This solution clearly yields a null value. €

Still, we may enhance the quality of such a relaxation by noticing that several additional constraints may be inserted to **MP\_EPC**:

* For any *j*, *Tj* + 1 ≥ *Tj*+*tj*;
*  *j Lj* ≥  *j ej*.

1. **Additional Constraints**.

For any j = 1,…, M, we set:

* Dj =  k = 0,…, j-1 tk + dj; D0 = d0;
* D\*j =  k = j+1,…, M tk + d\*j+1; D\*0 =  k = 1,…, M tk + d\* 1 ; D\*M = d\*M+1
* Minj = ⎡Dj/p⎤; Min0 = ⎡D0/p⎤;
* Maxj = N – 1 - ⎡D\*j/p⎤; Max0 = N – 1 - ⎡D\*0/p⎤;
* For any pair j1, j2 j1 < j2, j1,j2 = j1+1≤ j < j2 ej + \*j1+1 + j2.
* For any pair j1, j2, j1 < j2, INTj1,j2 = ⎡(j1+1≤ j < j2 tj + d\*j1+1 + dj2)/p⎤;

We say that 2 pairs (i1, j1) an (i2, j2) are *antagonistic* iff i1 < i2 and j1 > j2. A collection  of pairwise *antagonistic* pairs (i, j) is called an *antagonistic clique*.

We say that 2 pairs (i1, j1) an (i2, j2) are *time-inconsistent* iff (i2 – i1) ≤ INTj1,j2. A collection  of pairwise *time-inconsistent* pairs (i, j) is called an *time-inconsistent clique*.

***II.1. Simple Time Constraints***.

* For any j = 0, …, M: Tj+1 ≥ Tj + tj  + xj.(dj + d\*j – tj);

***II.2. Energy Constraints***.

We introduce additional variable Fj ≥ 0, j = 0,…, M+1= 0, with the meaning that Fj means the energy used by the vehicle from 0 to j. With F0 =0

* For any j = 0, …, M: Fj+1 ≥ Fj + ej  + xj.(j + \*j+1 – ej);
* For any j = 0,…, M: Fj – E0 ≤  k < j Lk;
* For any j = 1,…, M:  0 ≤ i ≤ Max(j-1)-1 Ri.zi ≥ Fj - E0 – H0;
*  0 ≤ i ≤ N-1 Ri.zi ≥ FM+1;
* For any j = 1,…, M:  0 ≤ i ≤ Max(j-1)-1 yi ≥ (Fj - E0 - H0)/CMP;
*  0 ≤ i ≤ N-1 yi ≥ (FM+1)/CMP;
* For any j1, j2, K such that j1,j2 > CVeh + K.CMP:  Minj1 ≤ i ≤ Maxj2 yi ≥ K;
* For any j1, j2 such that j1,j2 > CVeh:  j1 ≤ j ≤ j2-1 xj ≥ 1.

***II.3. Structural Constraints***.

* For any j, any i such that i < Minj or i > Maxj: Ui,j = 0;
* For any i, j1, j2: Ui,j1 + Ui+1,j2 ≤ 1;
* For any *antagonistic clique* :  (i,j) ∈  Ui,j ≤ 1.
* For any *time-inconsistent clique* :  (i,j) ∈  Ui,j ≤ 1.

1. **Separating the Structural Constraints**.

***III.1. Separating the Antagonistic*** ***Cliques***.

***SEPARE\_ANTAGO*:**

**Input**: Current vector U, rational.

**Output**: An antagonistic clique , which violates the antagonistic clique constraint. In case  in undefined, then no such a clique exists.

j <- 0; i <- N-1;

While j ≤ M+1 do

(i, j) < - 0;

For j1 ≤ j do

For i1 ≥ i do

If (i1 ≠ i) OR (j1 ≠ j) then

If Ui,j + (i1, j1) > (i, j) then

(i, j) <- Ui,j + (i1, j1);

Arg(i, j) <- (i1, j1).

If i ≥ 1 then i <- i – 1

Else

i <- N-1;

j <- j+

If (0, M+1) > 1 then  <- *Reconstruction* through Arg(0, M+1) else <- Undefined;

***III.2. Separating the Time-Inconsistent Cliques***.

**Preliminary**: *Weakening the time-inconsistency*.

Such as it has been defined, the time-inconsistency constraints seems difficult to separate. So we simplify them as follows:

* For any pair j1, j2, j1 < j2, we set INT\*j1,j2 = ⎣(j1 ≤ j < j2 tj)/p⎦.; If j1 = j2 then INT\*j1,j2 = 0.

So we separate the clique in the sense of INT\* (instead of INT), while making the assumption that:

* Inf j tj + Inf j dj + Inf j d\*j ≤ p.

In case this assumption is not satisfied, then our process works as an approximation.

Please also notice that constraint: *For any i, j1, j2: Ui,j1 + Ui+1,j2 ≤ 1*, is a specific case of the time-inconsistency constraint.

***SEPARE\_INCONSISTENT*:**

**Input**: Current vector U, rational.

**Output**: A time inconsistent clique , which violates the time-inconsistency clique constraint. In case  in undefined, then no such a clique exists.

j <- 0; i <- 0;

While j ≤ M+1 do

(i, j) < - 0;

For j1 ≤ j do

For i1 = i - (1+ INT\*j1,j2),…, i do

If (i1 ≠ i) OR (j1 ≠ j) then

If Ui,j + (i1, j1) > (i, j) then

(i, j) <- Ui,j + (i1, j1);

Arg(i, j) <- (i1, j1).

If i ≤ N-2 then i <- i + 1

Else

i <- 0;

j <- j+

If (N-1, M+1) > 1 then  <- *Reconstruction* through Arg(N-1, M+1) else <- Undefined;