RAIRO-Oper. Res. 55 (2021) 2359–2383
https://doi.org/10.1051/ro/2021094

RAIRO Operations Research
www.rairo-ro.org

PIPE-LINING DYNAMIC PROGRAMMING PROCESSES TO SYNCHRONIZE BOTH THE PRODUCTION AND THE CONSUMPTION OF ENERGY

Fatiha Bendali, Eloise Mole Kamga, Jean Mailfert, Alain Quilliot* and Hélène Toussaint

Abstract. Synchronizing heterogeneous processes remains a difficult issue in Scheduling area. Related ILP models are in trouble, because of large gaps induced by rational relaxation. We choose here to deal with it while emulating the interactions which take place between the various players of such heterogeneous processes, and propose a pipe-line decomposition of a dynamic programming process designed in order to schedule energy production and energy consumption

Mathematics Subject Classification. 90-08, 90C39.

Received September 30, 2020. Accepted June 17, 2021.

1. Context and state of the art

Efficiently synchronizing heterogeneous processes remains a difficult issue when it comes to scheduling and routing (see [13,17]). It arises for instance in the management of vehicle sharing systems, of drones and trucks in the context of urban logistics, and of industrial assembly processes. Integer Linear Programming (ILP) models are flawed by large gaps induced by the relaxation of the integrality constraint (the *Big M* problem). By the same way, designing *ad hoc Branch and Bound* schemes is difficult because of the lack of efficient bounding scheme. Besides, synchronization requirements increase the impact of uncertainty and put robustness at stake, making the efficiency of global heuristics difficult to check. A way to address those issues is to introduce flexibility and modularity in the design of algorithms and rely on *ad hoc* decomposition schemes in order to emulate the interaction mechanisms which allow distinct players to run a complex process in a decentralized way.

This synchronization issue tends to become a crucial one when it comes to energy management: The emergence of renewable energies (Photovoltaic, Wind, Hydrogen, ...) also means the emergence of local *in situ* producers which are at the same time consumers: factories, farms and even individual householders. Due to both market deregulation and emergent technologies, (see [1, 5, 6, 14]) this trend is forecast to have a major impact on Energy Economics. Key energy players are currently paying attention to it. A good example is provided by the activities of Labex IMOBS3 program in Clermont-Fd, France, devoted to *Innovative Mobility*. In the context of this project, we are currently involved into the control of a micro-plant for hydrogen production, which feeds autonomous vehicles with hydrogen fuel. But, while most hydrogen production is usually performed through power costly electrolysis processes, researchers rely here on solar power and photolysis [7,18,25], which make the

Keywords. Scheduling, dynamic programming, energy management.

LIMOS CNRS 6158, Labex IMOBS3, Clermont-Ferrand, France.
*Corresponding author: alain.quilliot@isima.fr

productivity of the process deeply depend on the intensity of solar illumination. According to this paradigm, the energy production/consumption process becomes endogenous, and performed according to a closed loop which requires high synchronization.

Few works have been addressing this specific issue of simultaneously managing energy production and consumption (see [1,14,17,29]). Most contributions are one sided and focus on either the consumer or the producer point of view: many are related to electric or hybrid vehicles required to minimize their energy consumption (*Green VRP*, *Pollution-Routing Problem* and *Hybrid Vehicle Problem*: see [15, 16, 20–23, 26]), while trying to optimize recharge transactions submitted to time windows or shared access constraints (see [33–35]). Also, some authors address the issue of scheduling an industrial process (see [1, 5, 6, 14, 17, 24, 28, 30–32]) while taking into account temporal variations of the energy costs, access restrictions and environmental concerns. While most contributions are related to applications, we must mention several theoretical contributions which deal with complexity and approximation issues, for models which put at stake the cost of idleness and the impact of time dependencies (see [2–4, 9–12]). On another side, main energy producers are carrying systematic studies about the way to plan energy production (gas, electricity, dam or nuclear plant management), according to a big grain point of view, that means with the purpose of meeting large scale uncertain aggregated demands (see [5, 6, 19, 29, 36]).

Because of the IMOBS3 project, we deal here with the synchronous management of, on one side, a fleet of small electric vehicles provided with hydrogen power cells, and, on the other side, a *micro-plant* in charge of local hydrogen fuel production. Taken as a whole, the problem involves forecasting, safety management and scheduling. Still, since our purpose is to focus on algorithmic features of synchronization, we restrict ourselves to the last issue, and set a simplified EPC: *Energy Production and Consumption* model restricted to the case of one vehicle required to perform tasks according to a pre-fixed order while periodically going back to the *micro-plant* in order to refuel. The *micro-plant* has its own production/storage restrictions, and our goal is to synchronize both the *micro-plant* and the vehicle. This model is NP-Hard, and its ILP formulation ill-fitted to numerical processing. We first address it according to a purely centralized paradigm, and propose an exact integrated *Dynamic Programming Scheme* (DPS), which implements synchronization while linking together *production* and *vehicle* time spaces. This DPS allows us to state a *Polynomial Time Approximation Scheme* (PTAS) result and provides us with optimal solutions in case of not too large EPC instances. Still it remains computing time costly. Besides, full integration of vehicle and production induces a lack of flexibility which make difficult dealing the collaborative features of real life contexts. So we adopt a point of view which consists in addressing EPC while emulating those collaborative features: We split our EPC model into two independent sub-models, one related to the vehicles and the other related to the *micro-plant*, which we tie together through some kind of collaborative *Demander/Producer* interaction.

The paper is organized as follows: Section 2 describes the EPC: *Energy Production/Consumption Problem* model, as well as a *Demander/Producer* decomposition of EPC into two interacting VD: *Vehicle-Driver* and PM: *Production-Manager* models. Section 3 first proposes an exact dynamic programming *DPS-EPC* algorithm, which solves EPC while tying together *vehicle* and *production* time spaces and relying on filtering devices, and provides us with reference optimal values. Then Sections 4 and 5 study respectively VD and PM, from both theoretical and algorithmic point of view, while Section 6 designs a *pipe-line* collaborative algorithmic scheme which implements the *Demander/Producer* decomposition of Section 2. Section 7 is devoted to numerical experiments, which aim at evaluating the quality of the tradeoff error versus running time induced by this *pipe-line* scheme. Conclusion provides a hint at future work, devoted to the robustness issue and to inclusion of the vehicle tour as part of the decision.

2. The EPC problem: mathematical formulation and decomposition scheme

We consider here some vehicle which has to perform internal logistic tasks, while following a route Γ which starts from some *Depot* node and ends at the same place after going through stations $j = 1, \ldots, M$, according to this order. Start-node *Depot* has label 0 and End-node *Depot* has label $M + 1$. The time required by the

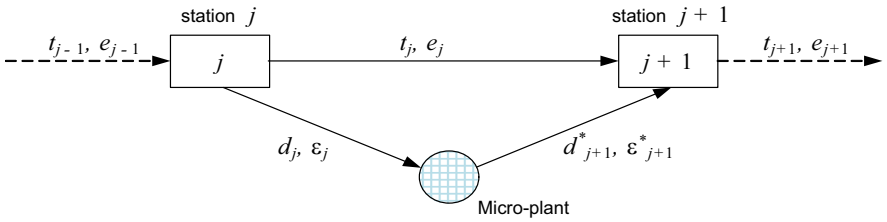

FIGURE 1. Time and energy coefficients for consecutive stations j and $j+1$.

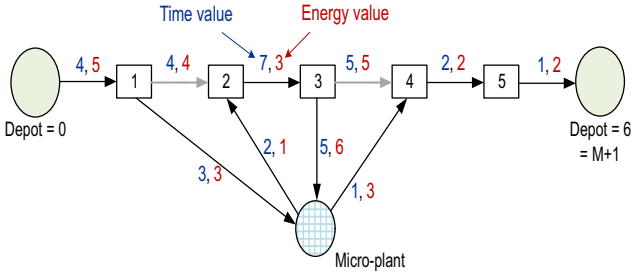

FIGURE 2. A vehicle trip, with its refueling transactions.

vehicle in order to go from j to $j+1$ is equal to t_j, taking into account the time spent by the vehicle in servicing j. The vehicle may leave *Depot* at time 0 and should be back no later than some time TMax.

Our vehicle is powered by hydrogen fuel. The capacity of its tank is denoted by C^{Veh} and we know, for any $j = 0, \ldots, M$, the hydrogen amount e_j required in order to move from station j to station $j+1$. The initial hydrogen load of the vehicle is denoted by E_0, and the vehicle is required to end its trip with at least the same hydrogen load. It comes that the vehicle must periodically refuel. Refueling transactions take place at a *micro-plant*, close to *Depot*: The time required by the vehicle in order to move from station j to the micro-plant (from the micro-plant to j) is denoted by $d_j(d^*_j)$; by the same way, the energy required in order to move from j to the micro-plant (from the micro-plant to j) is denoted by $\varepsilon_j(\varepsilon^*_j)$. Quantities d_j, d^*_j, t_j, e_j, ε_j, ε^*_j are strictly positive, satisfy the *Triangle Inequality* (see Fig. 1) and are such that $E_0 \geq \varepsilon_0$.

Figure 2 displays an example of a trip performed by the vehicle along station $Depot = 0, 1, 2, 3, 4, 5, 6 = Depot$, while refueling between station 1 and station 2, and next between station 3 and station 4.

On another side, the micro-plant produces H^2 *in situ* through photolysis and electrolysis. Resulting hydrogen is stored inside the micro-plant's tank, with capacity equal to C^{MP}. We suppose that the time space $\{0, \ldots, T\text{Max}\}$ is divided into N periods $P_i = [p.i, p.(i+1)]$, $i = 0, \ldots, N-1$, with $T\text{Max} = N.p$. We identify index i and period P_i. If the micro-plant is *active* at some time during period i, then it is active during the whole period i, and produces R_i hydrogen fuel units. For safety concerns, the vehicle cannot refuel while the micro-plant is producing: this assumption is due to the fact, in case of the experimental platform we are referring to, making hydrogen simultaneously arrive into the micro-plant's tank and leave it in order to be loaded into the vehicle's tank induces variations of pressure which raise safety issues. Besides any vehicle *refueling transaction* requires a whole period i. At time 0, the load of the micro-plant tank is $H_0 \leq C^{\text{MP}}$ and the micro-plant is not active. The same situation should hold at time TMax.

Producing hydrogen fuel has a cost, which may be decomposed into 2 components:

- A constant *activation cost* $Cost^F$, which is charged every time the micro-plant is activated.
- A *time-dependent* production cost $Cost^V_i$ which is independent on the amount of hydrogen really produced during period i and reflects the time-indexed prices charged by the electricity provider.

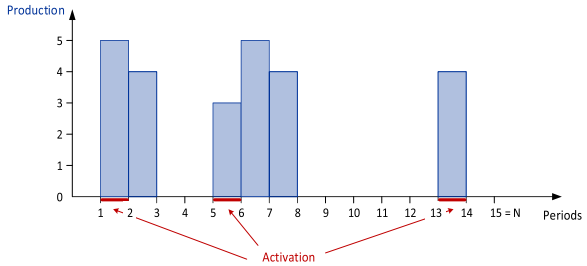

FIGURE 3. An example of micro-plant activity, with $N = 15$.

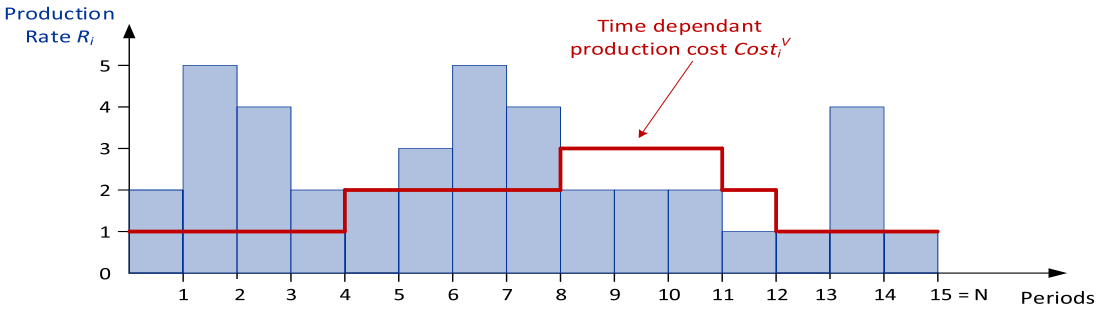

FIGURE 4. Production rates and time-dependent production costs for the micro-plant of Figure 3.

Then the *Energy Production/Consumption* (EPC) Problem consists in scheduling both the vehicle and the micro-plant in such a way that:

- The vehicle starts from $Depot = 0$, visits all stations $j = 1, \ldots, M$ and comes back to $Depot$ at some time $T \in [0, T\text{Max}]$, while moving to the micro-plant in order to refuel every time it is necessary;
- The micro-plant produces and stores in time the hydrogen fuel needed by the vehicle;
- Induced hydrogen production cost $Cost$ and vehicle ending time T are the smallest possible. We merge both costs into a unique one: $Cost + \alpha.T$, where α is some scaling coefficient.

Example 2.1. Figure 5 below synchronizes the vehicle and the micro-plant of Figures 2–4 in case $p = 2$, $E_0 = 8$, $H_0 = 4$, $T\text{Max} = 30$, $Cost^F = 7$, $C^{\text{MP}} = 15$, $C^{\text{Veh}} = 15$, $\alpha = 1$. In such a case, the vehicle refuels twice: the first refueling transaction, performed at period 4, involves 13 fuel units, and the second one, performed at period 12, involves 12 fuel units. Resulting tour ends at time 30 and production cost is $3.7 + 2 + 6 + 1 = 30$. It comes that resulting global cost is $30 + 30 = 60$.

The following Table 1 summarizes the input data for the EPC problem:

2.1. 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:**
  - $z_i \in \{0, 1\}$, with $i = -1, \ldots, N-1$: $z_i = 1 \sim$ the micro-plant is active during period i ($i = -1$ corresponds to a fictitious period);
  - $y_i \in \{0, 1\}$, with $i = 0, \ldots, N-1$: $y_i = 1 \sim$ the micro-plant is activated at the beginning of i;

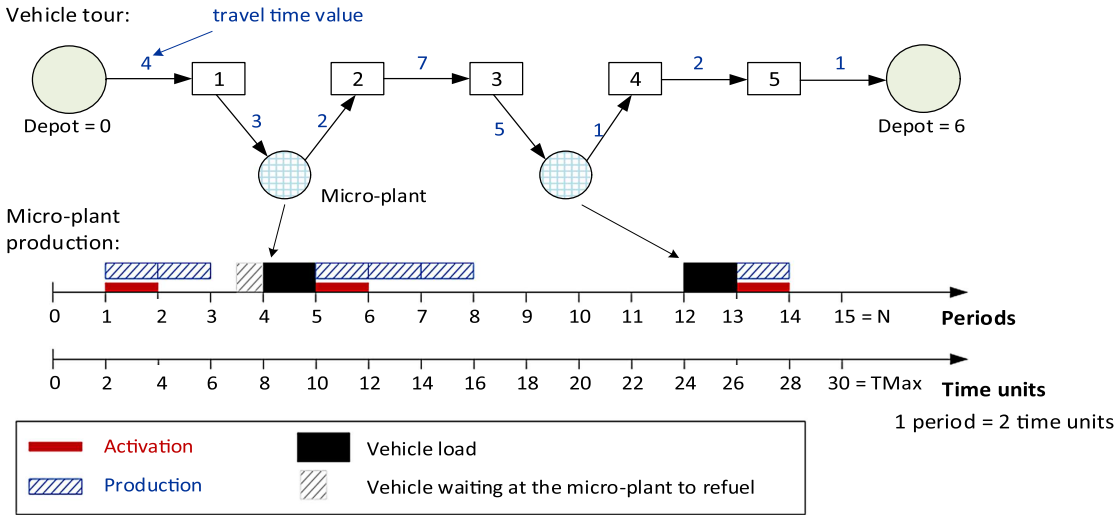

FIGURE 5. An EPC feasible solution related to Figures 2–4.

TABLE 1. Input data for the EPC problem.

Vehicle related input
M: number of stations (*Depot* excluded)
$\Gamma = (Depot = 0, 1, \ldots, M, Depot = M+1)$: vehicle tour (without refueling)
TMax: maximal time for the vehicle to achieve its tour
C^{Veh}: vehicle tank capacity
E_0: initial vehicle hydrogen load
$t_j \geq 0$, with $j = 0, \ldots, M$: required time to go from station j to station $j+1$
$d_j \geq 0$, with $j = 0, \ldots, M$: required time to go from station j to the micro-plant
$d_j^* \geq 0$, with $j = 0, \ldots, M$: required time to go from the micro-plant to station j
$e_j \geq 0$, with $j = 0, \ldots, M$: required energy to go from station j to station $j+1$
$\varepsilon_j \geq 0$, with $j = 0, \ldots, M$: required energy to go from station j to the micro-plant
$\varepsilon_j^* \geq 0$, with $j = 0, \ldots, M$: required energy to go from the micro-plant to station j
Micro-plant production related input
C^{MP}: micro-plant tank capacity
N: number of production periods
p: duration (in time units) of one production period
H_0: initial micro-plant hydrogen load
$Cost^F$: activation cost
$P_i = [p.i, p.(i+1)$ [with $i = 0, \ldots, N-1$: time interval related to production period i
$R_i \geq 0$, with $i = 0, \ldots, N-1$: production rate related to period i
$Cost_i^V \geq 0$, with $i = 0, \ldots, N-1$: production cost related to period i

- $V_i^{\text{Tank}} \geq 0$, with $i = 0, \ldots, N$: V_i^{Tank} 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;
- $\delta_i \in \{0, 1\}$, with $i = 0, \ldots, N-1$: $\delta_i = 1 \sim$ the vehicle is refueling during period i;
- $L_i^* \geq 0$, with $i = 0, \ldots, N-1$, with non negative integer values: in case $\delta_i = 1$, L_i^* is the quantity of hydrogen loaded by the vehicle during period i.

– **Vehicle variables:**
  - $x_j \in \{0,1\}$, with $j = 0,\ldots,M$: $x_j = 1 \sim$ the vehicle refuels while traveling from station j to station $j+1$;
  - $L_j \geq 0$, with $j = 0,\ldots,M$: if $x_j = 1$, $L_j =$ hydrogen quantity loaded by the vehicle while traveling from j to $j+1$;
  - $T_j \geq 0$, with $j = 0,\ldots,M+1$: $T_j =$ time when the vehicle arrives at j;
  - $T_j^* \geq 0$, with $j = 0,\ldots,M+1$: if $x_j = 1$, $T_j^* =$ time when the vehicle starts refueling while traveling from j to $j+1$;
  - $V_j^{\text{Veh}} \geq 0$, with $j = 0,\ldots,M+1$: $V_j^{\text{Veh}} =$ hydrogen load of the vehicle tank when the vehicle arrives in j.
– **Synchronization variables:** $U_{i,j} \in \{0,1\}$, with $i = 0,\ldots,N-1$, $j = 0,\ldots,M$: $U_{i,j} = 1 \sim$ 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
$\Sigma_{i=0,\ldots,N-1}\left(Cost^F.y_i + Cost_{i.}^V \cdot z_i\right) + \alpha.T_{M+1}$.
– **Production constraints:**
  - For any $i = 0,\ldots,N-1$: $y_i = 1 \rightarrow (z_i = 1 \wedge z_{i-1} = 0)$;
  - For any $i = 0,\ldots,N-1$: $z_i + \delta_i \leq 1$;
  - $z_{-1} = 0$;
  - $V_0^{\text{Tank}} = H_0$; $V_N^{\text{Tank}} \geq H_0$;
  - For any $i = 0,\ldots,N-1$: $V_i^{\text{Tank}} \leq C^{\text{MP}}$;
  - For any $i = 0,\ldots,N-1$: $V_{i+1}^{\text{Tank}} = V_i^{\text{Tank}} + z_i.R_i - \delta_i.L_i^*$.
– **Vehicle constraints:**
  - $T_0 = 0$; $V_0^{\text{Veh}} = E_0$; $V_{M+1}^{\text{Veh}} \geq E_0$;
  - For any $j = 1,\ldots,M+1$: $V_j^{\text{Veh}} \leq C^{\text{Veh}}$;
  - For any $j = 0,\ldots,M$: $V_j^{\text{Veh}} \geq \varepsilon_j$; (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 e_j and ε_j).
  - For any $j = 0,\ldots,M$: $T_{j+1} \geq (1-x_j).(T_j + t_j) + x_j.\left(T_j^* + p + d_{j+1}^*\right)$;
  - For any $j = 0,\ldots,M$: $T_j^* \geq T_j + d_j$;
  - For any $j = 0,\ldots,M$: $x_j = 0 \rightarrow V_{j+1}^{\text{Veh}} = V_j^{\text{Veh}} - e_j$;
  - For any $j = 0,\ldots,M$: $x_j = 1 \rightarrow V_{j+1}^{\text{Veh}} = V_j^{\text{Veh}} - \varepsilon_j - \varepsilon_{j+1}^* + L_j$;
  - $T_{M+1} \leq T\text{Max}$.
– **Synchronization constraints:**
  - For any $j = 0,\ldots,M$: $\Sigma_{i=0,\ldots,N-1} U_{i,j} = x_j$;
  - For any $i = 0,\ldots,N-1, \delta_i = \Sigma_{j=0,\ldots,M} U_{i,j}$;
  - For any $j = 0,\ldots,M, T_j^* = \Sigma_{i=0,\ldots,N-1} p.i.U_{i,j}$;
  - For any $i = 0,\ldots,N-1$: $L_i^* \leq \text{Inf}\left(V_i^{\text{Tank}}, \Sigma_{j=0,\ldots,M} U_{i,j}.\left(C^{\text{Veh}} + \varepsilon_j - V_j^{\text{Veh}}\right)\right)$; (E2)
  ((E2) expresses that load L_i^* cannot exceed neither the current load of the micro-plant tank nor the difference between C^{Veh} and the current load of the vehicle tank).
  - For any $j = 0,\ldots,M$: $L_j = \Sigma_{i=0,\ldots,N-1} U_{i,j}.L_i^*$.

2.2. A demander/producer decomposition scheme

We are now going to explain the collaborative how the above EPC model may be decomposed in such a manner which emulates the way decisions are going to be taken in (realistic) case decision is *collaborative*, which means that *production manage*r and *vehicle driver* are independent players, which are required to communicate. Main idea here is that a natural behavior of a decentralized *vehicle driver* is to move as if it were sure to get enough fuel every time he goes to the micro-plant, and to adapt itself to real context by waiting and eventually

delaying some moves. Conversely, a natural, market oriented behavior of the *production manager* will consist in adapting itself to demand, and cutting a trade-off between *Quality of Service* and production cost.

2.2.1. VD: Vehicle_Driver model

We forget here the restrictions related to hydrogen production and do as if the micro-plant were able to provide, at any time, the vehicle with as much as energy it needs. Then our goal is to fix the *Refueling Strategy* of the vehicle, that is the 0, 1-valued vector $x = (x_j, j = 0, \ldots, M)$ and the load vector $L = (L_j, j = 0, \ldots, M)$ of above model, which tell us at which stations j vehicle will refuel between j and $j+1$, and how much. Variables $T = (T_j, j = 0, \ldots, M+1)$, $T^* = (T_j^*, j = 0, \ldots, M+1)$ and $V^{\mathrm{Veh}} = (V^{\mathrm{Veh}}{}_j, j = 0, \ldots, M+1)$ may be considered as auxiliary variables, whose values derive from x and L in a natural way by noticing that:

- The vehicle never waits: it refuels as he arrives at the micro-plant, and keeps full speed meanwhile;
- At the last time it refuels, it does in such a way he is back at *Depot* with a load equal to E_0. At any other time, it achieves full tank.

Constraints are the *Vehicle Constraints* of EPC model of above Section 2.1. What remains to specify is the performance criterion. Of course, it has to include the $\alpha.T_{M+1}$ term. But it also has to involve a component which reflects what the economic cost of the *Refueling Strategy* (x, L) is likely to be. Since we do not know in advance what the *Production Strategy* is going to be, we introduce an auxiliary cost coefficient β, and consider that the economic cost of the *Refueling Strategy* (x, L) is the quantity $\beta.(\Sigma_j L_j.x_j)$. This coefficient is going to be a key component of the interaction between the vehicle (*demander*) and the micro-plant (*producer*). It comes that resulting VD: *Vehicle_Driver* model comes as follows:

VD: ***Vehicle_Driver*** Model: {Compute the *Refueling Strategy* (x, L), together with auxiliary variables $T = (T_j, j = 0, \ldots, M+1)$, $T^* = (T_j^*, j = 0, \ldots, M+1)$ and $V^{\mathrm{Veh}} = (V^{\mathrm{Veh}}{}_j, j = 0, \ldots, M+1)$, in such a way that:
- *Vehicle Constraints* of Section 2.1 are satisfied;
- The quantity $\alpha.T_{M+1} + \beta.(\sum_j L_j.x_j)$ is the smallest possible}.

2.2.2. The PM: Production_Manager model

Since the *production manager* is supposed to adapt itself to demand, we suppose here that he is provided with an information which reflects the fuel demand by the *vehicle driver*, such as it derives from an *ad hoc Refueling Strategy*. As a matter of fact, a *Refueling Strategy* (x, L) provides us with a number Q of *refueling transactions* performed by the vehicle, with related hydrogen loads $\mu_q, q = 1, \ldots, Q$, and with *optimistic* dates when those refueling transactions take place. But one understands that those dates are going to be the issue for a deal between the vehicle and the micro-plant. In order to bring flexibility to this deal, we shall use vector x in order to provide us with lower bounds $m_1, \ldots, m_Q$ and upper bound $M_1, \ldots, M_Q$ for the periods when the refueling transactions take place, as well as with minimal delays (*time lags*) between two such consecutive periods, due to the trip the vehicle is required to achieve between 2 consecutive *refueling transactions*. Resulting information, which we call a *Reduced Refueling Strategy*, will consist in:

- A number Q of *refueling transactions* performed by the vehicle; Those transactions are labeled $q = 1, \ldots, Q$, and supposed to take place according to this order;
- Related loads $\mu_q, q = 1, \ldots, Q$, of hydrogen which are loaded by the vehicle at any refueling transactions $q = 1, \ldots, Q$;
- Lower bounds $m_1, \ldots, m_Q$ and upper bounds $M_1, \ldots, M_Q$ for the period numbers $i_1, \ldots, i_Q \in \{0, \ldots, N-1\}$ when refueling transactions $q = 1, \ldots, Q$ will take place, as well as *Time Lag* coefficients $B_1, \ldots, B_Q$ which express constraints: For any $q = 1, \ldots, Q-1, i_{q+1} \geq i_q + B_q$.

Then our goal becomes to schedule the *Production Strategy* of the *micro-plant*, that means $\{0, 1\}$-vectors z and δ with indexation on $i = 0, \ldots, N-1$ with the same meaning as in Section 2.1, in such a way that:

- *Production Constraints* of Section 2.1 are satisfied (vector y comes as an auxiliary vector);

– The values i when $\delta_i = 1$ corresponds to Q periods $i_1, \ldots, i_Q$ which agree with lower bounds $m_1, \ldots, m_Q$, upper bounds $M_1, \ldots, M_Q$ and *Time Lag* coefficients $B_1, \ldots, B_Q$;
– Related values L_i^* of Section 2.1 EPC model corresponds to values $\mu_q, q = 1, \ldots, Q$.

As for the performance criterion, it clearly must involve the economic cost $\sum_{i=0,\ldots,N-1} (Cost^F.y_i + Cost^V{}_{i.z_i})$. But it also must contain some component which reflects the role of the term $\alpha.T_{M+1}$ of the objective function of the global EPC model. It comes that the *Production Strategy* (z, δ) should aim at minimizing the quantity $\alpha.p.i_Q + \sum_{i=0,\ldots,N-1}(Cost^F.y_i + Cost^V{}_{i.z_i})$. So resulting PM: *Production_Manager* model comes as follows:

PM: ***Production_Manager*** Model: {Compute the *Production Strategy* (z, δ), together with auxiliary variables $y = (y_i, i = 0, \ldots, N-1)$, $V^{\text{Tank}} = (V^{\text{Tank}}{}_i, i = 0, \ldots, N-1)$ and $L^* = (L_i^*, i = 0, \ldots, N-1)$, in such a way that:
– *Production Constraints* of Section 2.1 are satisfied;
– $\sum_i \delta_i = Q$: indices i such that $\delta_i = 1$ may be labeled $i_1 < \ldots < i_q < \cdots < i_Q$ in such a way that:
• For any $q, \mu_q = L^*_{iq}$;
• For any $q : m_q \leq i_q \leq M_q$;
• For any $q \leq Q-1, i_{q+1} - i_q \leq B_q$;
– The quantity $\alpha.p.i_Q + \sum_{i=0,\ldots,N-1}(Cost^F.y_i + Cost^V{}_{i.zi})$ is the smallest possible}.

2.2.3. The VD_PM decomposition

Then we see that EPC may be reformulated in a *collaborative* way as follows:

– **VD_PM Reformulation of EPC:** {**Fix parameter** β and compute a *Refueling Strategy* (x, L) in such a way that *Reduced Refueling Strategy* (Q, m, M, B, μ) yields a *Production Strategy* (z, δ) with minimal $\alpha.p.i_S + \sum_{i=0,\ldots,N-1}(Cost^F.y_i + Cost^V{}_i.z_i)$ cost}.

3. An exact integrated DPS_EPC algorithm

MP EPC model of Section 2.1 involves a large number of heterogeneous variables tied together by logical implications. Not surprisingly ILP libraries fail in computing optimal solutions as soon as N, M stop being small. It suggests that the EPC model is complex, which is confirmed by the following result:

Theorem 3.1. *EPC is NP-Hard.*

Proof. It is enough to suppose that $Cost^F$ is null, and do in such a way that the vehicle cannot start before production has been achieved. Then EPC happens to contain the Knapsack problem. □

Still, EPC lies at the (hypothetic) boarder between P and NP, and may be handled through a Dynamic Programming Scheme (DPS) *DPS-EPC*. Since both VD and PM sub-models underlie two distinct representations of the time (stations and real time $T \in [0, T\text{Max}]$ in the case of the vehicle, periods in the case of the micro-plant), the key point in this scheme lies on the way we link together (*synchronization device*) related time spaces. This *synchronization device* relies on two components:

– **Relative positioning relations $\ll$ and $==$ between a real time value T and a period index i:** for any period $i = 0, \ldots, N-1$, and any real time value $T \in \{0, \ldots, T\text{Max}\}$, we set:
• $T \ll i$ if $T < p.i$; $T \gg i$ if $T \geq p.(i+1)$;
• $T == i$ if $p.i \leq T < p.(i+1)$.
– ***Time* and *State* Spaces of the DPS scheme:** then we define the *time space* of our DPS as the set Δ of *time pairs* $(i, j), i = 0, \ldots, N, j = 0, \ldots, M+1$, provided with its standard partial ordering. For any such a *time pair* (i, j), a related *state* is a 4-uple $s = (Z, T, V^{\text{Tank}}, V^{\text{Veh}})$, with the meaning:
• $Z = 1 \sim$ the micro-plant is active at the end of period $i-1$, that means at time $p.i$.

- V^{Tank} and V^{Veh} are respectively the loads of the *micro-plant* tank at the beginning of period i and the load of the vehicle tank when it arrives at station j.
- T is a time value in $0, \ldots,$ TMax whose meaning derives from its position with respect to period i according to relations $==$ and $\ll$:
  - If $T \gg i$, then the vehicle is on the road to j, which it shall reach at time T;
  - If $T \ll i$, then the vehicle is between j and the micro-plant, possibly waiting for being refueled;
  - If $T == i$, then the vehicle is at j, and decides between keeping on to $j+1$ or moving to the micro-plant.

We derive from this *synchronization device* the other components of *DPS_EPC* in a natural way:

– **Decisions, Preconditions, Transitions and Costs:** a decision D related to a *time pair* (i, j) and a *state* $s = \left(Z, T, V^{\text{Tank}}, V^{\text{Veh}}\right)$ is a 3-uple $D = (z, x, \delta)$ in $\{0, 1\}^3$, with the meaning:
- $z = 1 \sim$ the micro-plant decides to produce during period i;
- x refers to a decision taken only when $T == i$: in such a case, $x = 0$ means that the vehicle moves from station j to station $j+1$ without refueling; $x = 1$ means that it refuels at the *micro-plant* while traveling from j to $j+1$.
- $\delta = 1 \sim$ the vehicle is located at the *micro-plant* and decides to refuel during period i, forbidding the micro-plant to be active during this period. It requires $T \ll i$ and $p.i - T \geq d_j$.

Decision is taken at the end of period $i-1$. For any *time pair* (i, j) and *state* $s = \left(Z, T, V^{\text{Tank}}, V^{\text{Veh}}\right)$, no more than 4 decisions D are feasible:

- **1st case:** $T \gg i$. The only choice is about z. Setting $z = 1$ requires $V^{\text{Tank}} + R_i \leq C^{\text{MP}}$. In any case, we shift from (i, j) to $(i+1, j)$, resulting *state* is $s' = \left(z, T, V^{\text{Tank}} + z.R_i, V^{\text{Veh}}\right)$ and related *transition cost* is $z.(Cost^F.(1 - Z) + Cost^V_{i.})$.
- **2nd case:** $T \ll i$ and $p.i - T < d_j$. Then the vehicle is moving from j to the micro-plant, and it cannot refuel yet. The only choice is about z, with preconditions and costs as in the first case.
- **3rd case:** $T \ll i$ and $p.i - T \geq d_j$. Then, we have 3 possibilities:
  - **Producing during period *i*:** $z = 1$; $\delta = 0$. It requires $V^{\text{Tank}} + R_i \leq C^{\text{MP}}$ and $p.(i+1) + p + d^*_{j+1} + \Sigma_{k \geq j+1} t_k \leq T$Max. Resulting states and costs are as in the first case.
  - **Refueling during period *i*:** $z = 0$; $\delta = 1$. It requires $\varepsilon_{j+1} + \varepsilon^*_{j+1} \leq \text{Inf}\left(C^{\text{Veh}}, V^{\text{Tank}} + V^{\text{Veh}} - \varepsilon_j\right)$. We shift from (i, j) to $(i+1, j+1)$ and resulting *state* is $s' = (0, p.(i+1) + d^*_{j+1}, V^{\text{Tank}} - \text{Inf}\left(V^{\text{Tank}}, C^{\text{Veh}} - V^{\text{Veh}} + \varepsilon_j\right),\ \text{Inf}\left(C^{\text{Veh}} - \varepsilon^*_{j+1}, V^{\text{Tank}} + V^{\text{Veh}} - \varepsilon_j - \varepsilon^*_{j+1}\right))$. *Transition cost* is $\alpha.(p.(i+1) + d^*_{j+1} - T)$. Notice that if it is possible for the vehicle to come back to *Depot* without refueling anymore, above formula must be modified in order to make appear that the vehicle only refuels what it needs in order to be back to *Depot* with a load equal to E_0.
  - **Doing nothing during period *i*:** $z = 0$; $\delta = 0$. We shift from (i, j) to $(i+1, j)$, resulting *state* is $s' = \left(0, T, V^{\text{Tank}}, V^{\text{Veh}}\right)$, and *transition cost* is null.
- **4th case:** $T == i$. Then we have 4 possibilities:
  - **Producing during period *i* and keeping the vehicle towards *j* + 1:** $z = 1, x = 0$. It requires $V^{\text{Tank}} + R_i \leq C^{\text{MP}}$ and $V^{\text{Veh}} - e_j - \varepsilon_{j+1} \geq 0$ and $T + t_j \gg i$. Then we shift from (i, j) to $(i+1, j+1)$, resulting *state* is $s' = \left(0, T + t_j, V^{\text{Tank}} + R_i, V^{\text{Veh}} - e_j\right)$ and *transition cost* is $(Cost^F.(1 - Z) + Cost^V_{i.}) + \alpha.t_j$.
  - **Not producing during period *i* and keeping the vehicle towards *j*+1:** $z = 0, x = 0$. It requires $V^{\text{Veh}} - e_j - \varepsilon_{j+1} \geq 0$. Then, if $T + t_j \gg i$, we shift from (i, j) to $(i+1, j+1)$, resulting *state* is $s' = (0, T + t_j, V^{\text{Tank}}, V^{\text{Veh}} - e_j)$ and *transition cost* is $\alpha.t_j$, else we shift from (i, j) to $(i, j+1)$, with same resulting *state* and *transition cost*.
  - **Not Producing during period *i* and moving the vehicle to the micro-plant:** $z = 0, x = 1$. It requires $\text{Sup}(p.(i+1), T + d_j) + d^*_{j+1} + p + \Sigma_{k \geq j+1} t_k \leq T$Max. Then we shift from (i, j) to $(i+1, j)$, resulting *state* is $s' = (0, T, V^{\text{Tank}}, V^{\text{Veh}})$ and *transition cost* is null.

- **Producing during period i and moving the vehicle to the micro-plant:** $z = 1, x = 1$. It requires $V^{\text{Tank}} + R_i \leq C^{\text{MP}}$ and $\text{Sup}(p.(i+1), T + d_j) + d^*_{j+1} + p + \Sigma_{k \geq j+1} t_k \leq T\text{Max}$. We shift from (i, j) to $(i+1, j)$ resulting *state* is $s' = (1, T, V^{\text{Tank}} + R_i, V^{\text{Veh}})$ and *transition cost* is $\left(Cost^F.(1 - Z) + Cost^V_i\right)$.

– **Initial and final *states*:** initial *state* corresponds to *time pair* $(0, 0)$ and 4-uple $s_0 = (0, 0, H_0, E_0)$. Final *state* corresponds to any *time pair* $(i \leq N, M+1)$, and any 4-uple $(Z, T \leq T\text{Max}, V^{\text{Tank}} \geq H_0, V^{\text{Veh}} \geq E_0)$.

– **Bellman equations:** with every *time pair* (i, j) and any *state* $s = (Z, T, V^{\text{Tank}}, V^{\text{Veh}})$, we associate its Bellman value W (smallest cost of a sequence of transitions from initial state s_0 at time $(0, 0)$ to state s at time (i, j). Then we implement our DPS framework according to a forward driven strategy: For any current *time pair* (i, j), $S(i, j)$ denotes the state subset computed with respect to (i, j); then we scan related state subset $S(i, j)$, and for any such a *state* $s = (Z, T, V^{\text{Tank}}, V^{\text{Veh}})$, we generate related *feasible* (in the sense of above requirements) *decision* set $\text{Dec}((i, j), s)$. For every $D = (z, x, \delta)$ in $\text{Dec}((i, j), s)$, we generate resulting *time pair* (i', j') and *state* $s' = (Z', T', V^{\text{Tank}'}, V^{\text{Veh}'})$, together with the value $W + \text{CT}$, where CT means the *transition cost* induced by D:
  - If s' does not appear yet in $S(i', j')$ then we insert s' into $S(i', j')$, together with value $W + \text{CT}$;
  - If s' appears in $S(i', j')$ with a value W^*, then if $W^* > W + \text{CT}$, then $W + \text{CT}$ becomes the value associated with (i', j') and s' else we discard s'.

We denote by *DPS_EPC* the DPS algorithm designed this way.

3.1. Filtering through rounding: a PTAS result

The number of *states* that an execution *DPS_EPC* creates significantly increases with M and N. Still, if we suppose that all EPC parameters are integral and that $T\text{Max}$, C^{MP} and C^{Veh} are bounded by polynomial functions of N and M, then one may check *DPS_EPC* becomes time-polynomial. It suggests that it should be possible to state some *Polynomial Time Approximation Scheme* (PTAS) result. As usual, such a result will rely on a rounding scheme:

Rounding *DPS_ECP* states: L and $n = a_0 + a_1.2 + \ldots + a_q.2^q$, being 2 integers, $a_i \in \{0, 1\}$, we first set:

– If $q \leq L$ then $Round(n, L) = n$ else $Round(n, L) = a_{q-K}.2^{q-L} + \ldots + a_q.2^q$;

– If $q \leq L$ then $Round^*(n, L) = n$ else $Round^*(n, L) = a_{q-K}.2^{q^-L} + \cdots + a_q.2^q + 2^{q-L}$.

We say that two integers n and m are equivalent *modulo the L largest bits* if $Round(n, L) = Round(m, L)$. By extension, 2 states $s_1 = (Z_1, T_1, V_1^{\text{Tank}}, V_1^{\text{Veh}})$ and $s_2 = (Z_2, T_2, V_2^{\text{Tank}}, V_2^{\text{Veh}})$ are equivalent *modulo the L largest bits* if $Z_1 = Z_2$, $Round(T_1, L) = Round(T_2, L)$, $Round(V_1^{\text{Tank}}, L) = Round(V_2^{\text{Tank}}, L)$, $Round(V_1^{\text{Veh}}, L) = Round(V_2^{\text{Veh}}, L)$. As for the $Round^*$ function, it will allow us to ease capacity and initial state constraints.

Turning *DPS_EPC* algorithm into a parametrized polynomial time algorithm *DPS_EPC*(K)

We set $L_0 =$ Smallest integer L such that $2^L \geq N + M + 1$, and, for any integer $K \geq 1$, we proceed as follows:

(1) We ease initial values H_0 and E_0 by replacing them respectively by $Round^*(H_0, K + 1)$ and $Round^*(E_0, K + 1)$. By the same way, we relax the time capacity constraint by replacing $T\text{Max}$ by $Round^*(T\text{Max}, K)$ and capacities C^{MP} and C^{Veh} respectively by $Round^*(C^{\text{MP}}, K)$ and $Round^*(C^{\text{Veh}}, K)$: this means we check the feasibility of any decision $D = (z, x, \delta)$ with respect to time capacity $Round^*(T\text{Max}, K)$ and hydrogen capacities $Round^*(C^{\text{MP}}, K)$ and $Round^*(C^{\text{Veh}}, K)$.

(2) **We extend the notion of *state***, by considering that any *extended state* s is a 5-uple $(Z, \Omega, T, V^{\text{Tank}}, V^{\text{Veh}})$ associated with a time pair (i, j), where Ω is a 3-valued variable which tells us whether $T \ll i (\Omega = 0), T == i (\Omega = 1), (T \gg i) \wedge (T + d_j > p.i) (\Omega = 2), (T \gg i) \wedge (T + d_j \leq p.i) (\Omega = 3)$; It comes that the notion of *equivalence modulo the L largest bits* is extended the same way: $s_1 = (Z_1, \Omega_1, T_1, V_1^{\text{Tank}}, V_1^{\text{Veh}})$ and $s_2 = (Z_2, \Omega_2, T_2, V_2^{\text{Tank}}, V_2^{\text{Veh}})$ requires $\Omega_1 = \Omega_2$. This means that relative positioning of T and i through relations $\ll$, $\gg$ and $==$ acts as an explicit *state* variable, in order to compensate the fact that those relations may be perturbed by rounding effects, and that we perform all tests which involves relations $\ll$, $\gg$ and $==$ while referring to Ω.

(3) We do in such a way that, at any time, $S(i, j)$ does not contain 2 *extended states* s_1, s_2, together with values W_1, W_2 such that respectively s_1 and s_2, as well as W_1 and W_2, are equivalent *modulo the* $K + 2.L_0$ *largest bits*. We give priority to *state* s_q, $q = 1, 2$, related to the smallest W_q value.

Then we may state:

Theorem 3.2 (Polynomial Time Approximation Scheme)**.** *K being fixed, DPS_EPC (K) is time-polynomial. Besides, for any value* $\varepsilon > 0$*, we may choose K large enough in such a way that in case EPC admits an optimal solution with value* W^{Opt}*, then DPS_EPC (K) yields a solution which is feasible with regards to initial values* $(1+\varepsilon/2)\cdot H_0$ *and* $(1+\varepsilon/2)\cdot E_0$*, capacity values* $(1+\varepsilon)\cdot C^{\mathrm{MP}}$*,* $(1+\varepsilon)\cdot C^{\mathrm{Veh}}$ *and* $(1+\varepsilon)$*. TMax and whose cost value is no larger than* W^{Opt}*.*

Proof. It is a mere consequence of the way we have been implementing our rounding strategy: Priority given in 3) to state s_q, $q = 1, 2$, with smallest W_q value, allows us to compute a solution with cost value no larger than W^{Opt}; Augmenting (in 1) initial values H_0 and E_0 together with capacities C^{MP}, C^{Veh} and TMax allows us to keep this solution inside feasibility in the sense of Theorem 3.2. A key point lies in the way we extend in 2) the *state* notion through the introduction of component Ω: it allows us to keep well-fitted feasible *DPS_EPC*(K) decisions while erasing the impact of the rounding devices. □

3.2. Logical filtering devices

In spite of above result, the number of *state*s becomes too large when N and M increase. In order to reduce it, we may apply the following *Strong Dominance* Rule:

– For any *time pair* (i, j), if *states* $s_1 = (Z_1, T_1, V_1^{\mathrm{Tank}}, V_1^{\mathrm{Veh}})$ and $s_2 = (Z_2, T_2, V_2^{\mathrm{Tank}}, V_2^{\mathrm{Veh}})$ given together with values W_1 and W_2 are such that: $W_1 \leq W_2$; $T_1 \leq T_2$; $Z_1 \geq Z_2$; $V_1^{\mathrm{Tank}} \geq V_2^{\mathrm{Tank}}$; $V_1^{\mathrm{Veh}} \geq V_2^{\mathrm{Veh}}$, then *kill* s_2.

This rule has little filtering power. But other devices may be implemented. We distinguish:

– **Logical filtering rules:** at any *time pair* (i, j) during the *DPS_EPC* process, we anticipate that it will not be possible to extend current state $s = (Z, T, V^{\mathrm{Tank}}, V^{\mathrm{Veh}})$ into a feasible schedule, either because there is not enough time left (*Makespan Based* rules) or because it will not be possible to achieve required energy production (*Energy Based* rules), and so we kill s.
– **Quality based filtering rules:** at *time pair* (i, j) we check that the cost of any schedule consistent with current state $s = (Z, T, V^{\mathrm{Tank}}, V^{\mathrm{Veh}})$, will be at least equal to the cost of some current feasible schedule.

3.2.1. Logical filtering rules

For any period $i = 0, \ldots, N-1$, we denote by *Prod_Max*(i) the maximal quantity $\Sigma_{k \geq i}\ R_k$ that the micro-plant can produce from time $p.i$ on. Also, for any station $j = 0, \ldots, M$, we get a rough estimation of both energy and time required by the vehicle in order to return from j to *Depot* by applying the following process:

Bound_Fuel (j) ***Procedure*****:**
Refuel $\leftarrow \Sigma_{k \geq j}\ e_k + E_0$; Not *Stop*; *Refuel_Number* $\leftarrow 0$; Label the quantities $\sigma_k = \varepsilon_k + \varepsilon_k^* - e_k$, $k = j, \ldots, M$ by increasing order $\sigma_1 < \ldots < \sigma_{M+1-j}$;
While Not *Stop* **do**;
Refuel_Number_Aux $\leftarrow \lfloor$*Refuel* $/C^{\mathrm{Veh}} \rfloor$;
If *Refuel_Number_Aux* = *Refuel_Number* then *Stop* **Else**
$Refuel \leftarrow Refuel + \Sigma_{q=Refuel\_Number,\ldots,Refuel\_Number\_Aux}\sigma_q$; $Refuel\_Number \leftarrow Refuel\_Number\_Aux$;

Let us respectively denote by $Refuel_j$ and $Refuel\_Number_j$ the quantities *Refuel* and *Refuel_Number* obtained at the end of *Bound_Fuel*(j). The vehicle will have to load at least $Refuel_j - V^{\mathrm{Veh}}$ fuel units through at least $Refuel\_Number_j$ *refueling transactions* in order to achieve its tour from j. By taking into account those $Refuel\_Number_j$ transactions, we easily derive, for any j, a lower bound Δ_j on the time required from j to *Depot*. This allows us to implement the 2 following *logical filtering* rules, which may be applied to any *time pair* (i, j) and any related state $s = (Z, T, V^{\mathrm{Tank}}, V^{\mathrm{Veh}})$:

(1) *Makespan Based* filtering rule: if $(\Delta_j \geq T\text{Max} - T + 1)$ then *kill* s, since there is not enough time left for the vehicle to achieve its trip.
(2) *Energy Based* filtering rule: if $Refuel_j > V^{\text{Veh}} + Prod\text{-}Max(i) + V^{\text{Tank}}$ then *kill* s, since there won't be enough energy for the vehicle to achieve its trip.

3.2.2. Quality based filtering rules

For any period $i = 0, \ldots, N-1$, any micro-plant state value Z of the micro-plant at the end of period $i-1$, and any energy amount V, we pre-compute (through backward driven DPS), the minimal cost $Cost\_Min(i, V, Z)$ required from the *micro-plant* to produce V energy units between time $p.i$ and time TMax, while starting with state Z at the beginning of period i:

- $Cost\_Min(N, V, Z) = 0$ if $V = 0$ and undefined else;
- If $i \leq N-1$ then $Cost\_Min(i, V, Z) = \text{Inf}[Cost\_Min(i+1, V, 0), Cost\_Min(i+1, V - R_i, 1) + (Cost^F.(1 - Z) + Cost^V_i.]$.

Let us consider now some *time pair* (i, j) and some related state $s = (Z, T, V^{\text{Tank}}, V^{\text{Veh}})$ with value W. We deduce from above pre-computation a lower bound LB for the best EPC value which may be derived from (i, j) and s by setting: $\text{LB} = \text{LB}((i, j), s) = \alpha.\Delta_j + Cost\_Min(i, (Refuel_j - V^{\text{Tank}})^+, Z)$, with Δ_j and $Refuel_j$ computed as in Section 3.2.1. This lower bound procedure LB enables us to get an initial EPC solution by applying the *DPS_EPC* scheme in a greedy way:

***GREEDY_EPC* Procedure:**
Initialization: $(i, j) < -(0, 0)$; $s < -(0, 0, H_0, E_0)$; Not *Fail*;
While s is not a final state and Not *Fail* **do**
Select feasible decision $D = (z, x, \delta)$ such that $Cost\_Transition((i, j), s, D) + \text{LB}((i_1, j_1), s_1)$ is minimal, where *time pair* (i_1, j_1) and *state* s_1 result from the application of D.

Let us denote by *Curr_Val* the cost value of the EPC solution computed by *GREEDY_EPC*. Then, for any *time pair* (i, j), any state $s = (Z, T, V^{\text{Tank}}, V^{\text{Veh}})$ given together with some W cost value, we may apply the rule:

(3) *Quality Based* filtering rule: if $W + \text{LB}((i, j), s) \geq Current\_Val$, then *kill state* s.

4. The *Vehicle_Driver* problem

Let us recall that the VD: *Vehicle_Driver* model was stated as follows:

VD: *Vehicle_Driver* Model: {Compute the *Refueling Strategy* (x, L), together with auxiliary variables $T = (T_j, j = 0, \ldots, M+1), T^* = (T^*_j, j = 0, \ldots, M+1)$ and $V^{\text{Veh}} = (V^{\text{Veh}}_j, j = 0, \ldots, M+1)$, in such a way that:
- *Vehicle Constraints* of Section 2.1 are satisfied;
- The quantity $\alpha.T_{M+1} + \beta.(\Sigma_j L_j.x_j)$ is the smallest possible}.

4.1. Complexity of VD

We are going to prove that VD is polynomial. However, we are still at the boarder between P and NP since:

Theorem 4.1. *If we impose any L_j to be either null or equal to some constant H, then VD becomes NP-Hard.*

Proof. For any j, let us set: $A_j = d_j + d^*_{j+1} - t_j$ and $B_j = H - \varepsilon_j - \varepsilon^*_{j+1} + e_j$. Let also set $C = \Sigma_j e_j$. Finally, let us suppose $C^{\text{Veh}} = +\infty, \beta = 0$ and $T\text{Max} = +\infty$. Then we see that x should be such that:

- $\Sigma_j x_j.A_j$ is the smallest possible;
- $\Sigma_j x_j.B_j \geq C$.

Under those assumptions, VD coincides with Knapsack. Conversely, any Knapsack instance may be turned this way into a VD instance. □

The *VD_Graph* graph: in order to deal with VD, we are going to make appear that it may be handled as a simple shortest path problem in the following oriented graph *VD_Graph*:

- Nodes of *VD_Graph* are:
  - A source s and a sink $p = ((M+1), E_0)$;
  - Pairs $(0, V), 0 \leq V \leq E_0$;
  - Pairs $(j, V), j = 1, \ldots, M$, where V is a non negative number, no larger than C^{Veh};
- Arcs of *VD_Graph* are:
  - Any arc $u = (s, (0, V))$, with non positive cost $D_u = (V - E_0)$;
  - Any arc $u = ((j, V), (j+1, V - e_j))$, with null cost D_u;
  - Any arc $u = ((j, \varepsilon_j), (j+1, V)), V \geq \varepsilon_{j+1}$ with cost $D_u = \alpha.(d_j + d^*_{j+1} - t_j + p) + \beta.(\varepsilon_j + \varepsilon^*_{j+1} - e_j)$.

The meaning of this construction comes through following Lemma 4.2:

Lemma 4.2. *Solving VD means searching for a shortest path from s to p in VD_Graph.*

Proof. We say that *Refueling Strategy* (x, L) satisfies the *Empty Tank* Hypothesis if:

- Every time but the first time the vehicle refuels, it arrives at the micro-plant with an empty tank;
- It comes back to *Depot* with a hydrogen load exactly equal to E_0.

Then we notice that an optimal *Refueling Strategy* (x, L) may be chosen in such a way it satisfies this *Empty Tank* Hypothesis. If (x, L) does not agree with the *Empty Tank* Hypothesis, and it arrives into the *micro-plant* with a non null load $\delta > 0$, then it is possible to make decrease former refueling transaction by δ, and augment current refueling transaction by $\delta' \leq \delta$, until cancelling related *refueling transaction* or making it *Empty Tank*. It comes that:

- Nodes of *VD_Graph* corresponds to the possible states of the vehicle when he performs his trip from *Depot* $= 0$ to *Depot* $= M+1$, while visiting stations $j = 1, \ldots, M$ and periodically refueling. In case the vehicle has never been refueling before $j, V \leq E_0$ means the hydrogen amount it is going to consume before refueling; Else it means the current hydrogen load of the vehicle at j.
- According to this, we understand the meaning of the arcs:
  - Arc $u = ((j, V), (j+1, V - e_j))$, with null cost D_u, means that the vehicle move directly from j to $j+1$;
  - Arc $u = ((j, \varepsilon_j), (j+1, V)), V \geq \varepsilon_{j+1}$ with cost $D_u = \alpha.(d_j + d^*_{j+1} - t_j) + \beta.(\varepsilon_j + \varepsilon^*_{j+1} - e_j)$, means that the vehicle move from j to $j+1$ while refueling at the micro-plant: the cost of such a move is the additional cost (energy + time) induced by this detour;
  - Arc $u = (s, (0, V))$, with $V \leq E_0$, and negative cost $D_u = (V - E_0)$, means the decision that the vehicle is going to take at the beginning of the process, when it is going to decide about its first *refueling transaction*: related trip from 0 until *Depot* is then going to require V hydrogen units, and negative cost $D_u = (V - E_0)$ corresponds to the fact that the vehicle arrives at the *micro-plant* with an excess load $(E_0 - V)$ which will be used later.

We deduce that any *Refueling Strategy* (x, L) which satisfies the *Empty Tank* Hypothesis gives rise to a path in *VD-Graph*, whose cost is exactly the cost of the *Refueling Strategy* (x, L), and conversely. □

As a matter of fact, we do not need the whole graph *VD_Graph* in order to handle VD. If we apply a backward driven Bellman algorithm, then we see that we only deal with the following node collection $X(j), j = 0, \ldots, M+1$, whose recursive definition comes as follows:

- $X(M+1)$ is reduced to $p = ((M+1), E_0)$; $X(M) = \{(M, \varepsilon_M), (M, E_0 + e_M)$;
- For $j = 0, \ldots, M-1, X(j) = \{(j, \varepsilon_j)\} \cup \{(j, V + e_j)$, for all $(j+1, V) \in X(j+1)$ such that $V + e_j \leq C^{\text{Veh}}\}$.

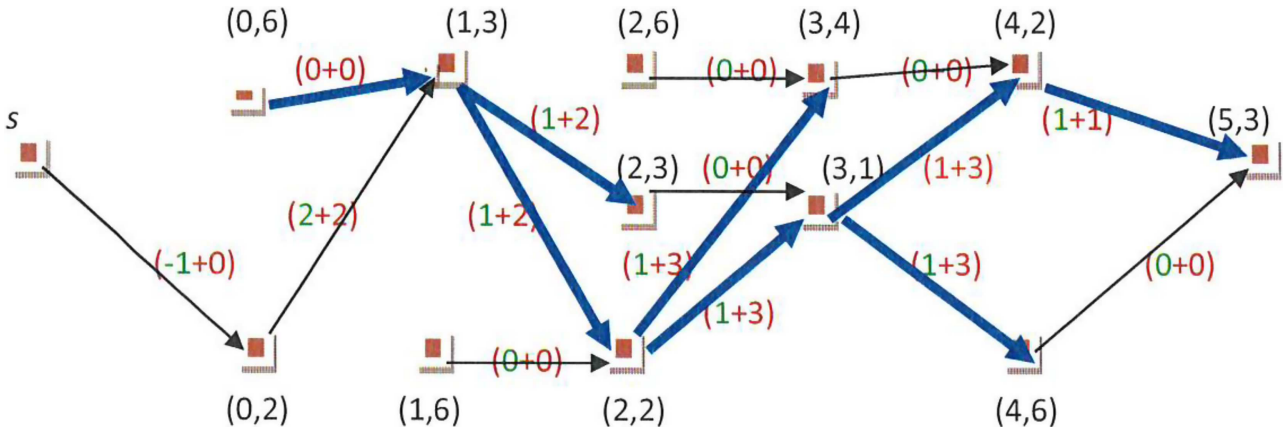

FIGURE 6. The Useful VD_Graph Subgraph.

If we set $X = \{s\} \cup (\cup_{j=0,\ldots M+1} X(j))$, then we call *Useful VD-Subgraph* the subgraph of *VD-Graph* induced by X.

Example 4.3. The *Useful VD_Subgraph* in case $M = 4$, $C^{\text{Veh}} = 6$, $E_0 = 3$, $\alpha = \beta = 1$, and values t, d, e, ε given by the table below:

j	0	1	2	3	4	$5 = 0$
e_j	3	4	2	2	3	*
$\varepsilon_j = \varepsilon_j^*$	2	3	2	1	2	2
t_j	4	5	3	2	6	*
$d_j = d_j^*$	3	3	4	2	3	4

Thick arrows represent here the moves of the vehicles which involve a refueling detour. For every arc, $(a+b)$ means the sum of respectively the energy and the time components of the cost of the arc (Fig. 6).

VD time-polynomiality: *VD_Graph* construction and Lemma 4.2 allow us to state:

Theorem 4.4. *VD can be solved in polynomial time.*

Proof. Because of Lemma 4.2 and the above construction of the *Useful VD_Subgraph*, we see that solving VD means searching a shortest path from s to p in the *Useful VD_Subgraph*. But, for any j, the cardinality of $X(j)$ does not exceed $M - j + 1$, and the number of arcs which connects $X(j)$ to $X(j+1)$ does not exceed Card($X(j)$ + Card($X(j+1)$). We conclude. □

4.2. VD dynamic programming handling

We deal with VD while computing a shortest path in the *Useful VD_Subgraph* according to a Bellman backward driven process. This leads us to the following *DPS_VD* dynamic programming algorithmic scheme:

The *DPS_VD* algorithm: Time, Space, Decisions and Transitions

- **Time Space:** in a natural way, related **Time Space** is the set $J = \{0, 1, \ldots, M, M+1\}$. We are going to scan this time space J backward.
- **State Space:** a state s associated with $j \in J$ is a pair $s = (V^{\text{Veh}}, T)$ which means the current load of the vehicle at j and a time value $T \leq T\text{Max}$, given together with a cost value W^{Veh} and a decision x°: T is the time (in the sense of real time) the vehicle will spend while moving from *state* s at station j until

state E_0 at time $M+1$; $W^{\text{Veh}} = \alpha.T + \beta.U$, where U is the energy to be wasted by the vehicle before the end of its trip; x° is the decision related to W^{Veh} through backward driven Bellman equations. Initial state, related to *time* value $M+1$, is going to be $(E_0, 0)$; Final states, related to $j = 0$, should be any pair $(V^{\text{Veh}} \leq E_0, T \leq T\text{Max})$. All pairs (j, s) should be nodes of the *Useful VD Subgraph*.

- **Decision Space:** a decision at time j is a binary number $x \in \{0,1\}$: $x = 0$ means a direct move from j to $j+1$ without refueling, while $x = 1$ means that a refueling detour through the *micro-plant* has to be performed before reaching $j+1$.
- **Backward driven strategy, transitions and costs:** we follow Theorem 4.4 and the *Useful VD Subgraph* construction, and so perform our DPS process according to a backward driven strategy. Notice that running our algorithm in case $\alpha = 0$ and $\beta = 1$, or $\alpha = 1$ and $\beta = 0$, provides us, for any j and any current load V^{Veh}, with the minimal time and energy amount required in order to achieve the vehicle tour from station j with load V^{Veh}. We may store this information in order to use it as a filtering tool for the handling of the *DPS_EPC* algorithm of Section 3. Backward transition from any state $s_1 = (V_1^{\text{Veh}}, T_1)$ at time $j+1$ induced by a decision x taken at time $j \geq 0$ comes as follows:
  - If $x = 0$, then resulting *state* $s = (V^{\text{Veh}}, T)$ associated with j is equal to $(V_1^{\text{Veh}} + e_j, T_1 + t_j)$. This transition requires $T \leq T\text{Max}$ and $V^{\text{Veh}} \leq C^{\text{Veh}}$; Its cost is 0.
  - If $x = 1$, then resulting *state* $s = (V^{\text{Veh}}, T) = (\varepsilon_j, d_j + p + d^*_{j+1} + T_1)$. This transition requires $T \leq T\text{Max}$ and $V^{\text{Veh}} \leq C^{\text{Veh}}$; Its cost is $\alpha.(d_j + p + d^*_{j+1} - t_j) + \beta.(\varepsilon_j + \varepsilon^*_{j+1} - e_j)$.
- **Bellman equations:** clearly, in case $j \geq 0$, and for any related (s, W^{Veh}) we should achieve:
  - $W^{\text{Veh}} = \text{Inf}_{x \in \{0,1\}, x\ feasible}\ (W_1^{\text{Veh}} + \text{Cost of the transition induced by } x)$, where W_1^{Veh} is the $W^{\ Veh}$ value related to resulting state s_1; Related Decision $v^\circ = Arg\ \text{Inf}$.
- **Optimal *Refueling Strategy*:** it comes by picking up the backward final state $(V^{\text{Veh}} \leq E_0, T \leq T\text{Max})$ related to $j = 0$, which provides us with the lowest $W^{Veh} - \beta.E_0$ value, and by following the stations $j = 0, \ldots, M$ according to decisions x°.

4.3. Retrieving a (primary) refueling strategy and a reduced refueling strategy

According to Section 4.2, we retrieve full vector x, together with load vector L, by applying the following process:

***Refueling_Strategy_Retrieval* procedure:**
$j < -0$; $V < -V^{\text{Veh}} \leq E_0$, such that related $T \leq T\text{Max}$ and provided with minimal $W^{\text{Veh}} - \beta.E_0$ value;
$T_0 < -$ Related T value; $Q < -0$; $x^\circ < -$ Related decision; $V\_Aux < -E_0 - V^{\text{Veh}}$;
$T\text{Sup}_0 < -T\text{Max} - T_0$; $T\text{Inf}_0 < -0$;
While $j \leq M$ **do**
  Let V_1, T_1 be time and energy values resulting from x°;
  If $x^\circ = 0$ **then**
    $x_j < -0$; $j < -j+1$; $V < -V_1$; $x^\circ < -$ Decision related with $(j+1, V^{\text{Veh}})$;
  Else
    $Q < -Q+1$; $\mu_Q = L_j < -(V_1 + \varepsilon^*_{j+1} - V\_Aux)^+$; $V\_Aux < -(V\_Aux - L_j)^+$; $x_j < -1$;
    $T\text{Sup}_Q < -T\text{Max} - (T_1 + d^*_{j+1} + p)$; $\Delta_{Q-1} < -T\text{Sup}_Q - T\text{Sup}_{Q-1}$; $T\text{Inf}_Q < -T\text{Inf}_{Q-1} + \Delta_{Q-1}$;
  $j < -j+1$;

Above process yields what we call a *Primary Refueling Strategy*, that means vectors x and L, together with a number Q and $\{0, \ldots, Q\}$ indexed vectors $T\text{Inf}$, $T\text{Sup}$ and μ, with the following meaning:

- Q is the number of refueling transactions performed by the vehicle;
- For $q = 1, \ldots, Q$:
  - μ_q is the quantity of fuel which is loaded during the refueling transaction q;
  - $T\text{Inf}_q$ ($T\text{Sup}_q$) is the earliest (latest) time when this refueling transaction may start (we consider $q = 0$ as referring to a fictitious refueling transaction, with $T\text{Inf}_0 = 0$ and $T\text{Sup}_0 = T\text{Max} - T$, where T is the time value associated with an optimal initial state);

- Δ_q is the minimal delay between the date of the q^{th} refueling transaction and the date of the $(q+1)$th refueling transaction (if $q = 0$ then $\Delta_0 = T\text{Inf}_1$).

***Reduced Refueling Strategy*:** in order to synchronize this *Refueling Strategy* with the hydrogen production process, we need to turn (routine process) values TInf, TSup and Δ in terms of periods $i = 0, \ldots, N-1$. By doing this we derive what we call a *Reduced Refueling Strategy*, that is:

- A number Q of *refueling transactions*;
- Lower bounds $m_0, \ldots, m_Q$ and upper bounds $M_0, \ldots, M_Q$ for the period numbers $i_0, \ldots, i_Q \in \{0, \ldots, N-1\}$ when the refueling transactions take place, as well as *Time Lag* coefficients $B_0, \ldots, B_Q$ which reflects the following constraints: For any $q = 0, \ldots, Q-1, i_{q+1} \geq i_q + B_q$ (we consider $i_0 = 0$ as a fictitious refueling transaction, which require 0 period).
- Loads μ_q = quantities of H^2 which is loaded for every value $q = 1, \ldots, Q$.

5. The *Production_Manager* (PM) problem

We suppose now that we are provided with a *Reduced Refueling Strategy* Q, $m = (m_0, \ldots, m_Q), M = (M_0, \ldots, M_Q)$, $B = (B_0, \ldots, B_Q)$ and $\mu = (\mu_q, q = 1, \ldots, Q)$ as we just defined above. Then solving PM consists in scheduling the activity of the *micro-plant*, that means computing $\{0, 1\}$ vectors z and δ with indexation on $i = 0, \ldots, N-1$ as in such a way that:

- *Production Constraints* of Section 2.1 are satisfied (vector y comes as an auxiliary vector);
- The values i when $\delta_i = 1$ corresponds to Q periods $i_1, \ldots, i_Q$ which comply with lower bounds $m_1, \ldots, m_Q$, upper bounds $M_1, \ldots, M_Q$ and *Time Lag* coefficients $B_1, \ldots, B_Q$;
- Related values L_i^* corresponds to values $\mu_q, q = 1, \ldots, Q$.
- The micro-plant ends with a hydrogen load at least equal to the quantity H_0 it started with;
- The quantity $\alpha.p.i_Q + \Sigma_{i=0,\ldots,N-1}\ (Cost^F.y_i + Cost^V_{i.}.z_i)$ is the smallest possible.

Notice that PM is NP-Hard, since it clearly may be viewed as an extension of the Knapsack problem with release and due dates for the delivery of the production. As a matter of fact, it is enough to set $Q = 1, m_1 = M_1 = N-1$ to make PM coincide with Knapsack.

5.1. Dealing with PM through dynamic programming

In order to deal with this *Production* problem, we apply, as for the general EPC Problem, a forward driven dynamic programming *DPS_PM* algorithm. But the size of related *Time, State*, and *Decision* spaces are significantly smaller. As a matter of fact those components come as follows:

The *DPS_PM* algorithm: Time, Space, Decisions and Transitions

- **Time Space:** the *Time Space* (in the DPS sense) is the set $I = \{0, \ldots, N\}$, scanned in the forward sense.
- **State Space:** for any $i = 0, \ldots, N$, a state is a 4-uple $E = (Z, V^{\text{Tank}}, Rank, Delay)$, with $Rank = 0, \ldots, Q$:
  - $Z = 1$ means that the micro-plant is active at the end of period $i-1$.
  - V^{Tank} means the current load of the *micro-plant* tank at the beginning of period i.
  - $Rank \in 0, \ldots, Q$ means that the *Rank*th refueling transaction has been performed and that we are waiting for the $(Rank+1)$th refueling transaction.
  - *Delay* means the difference between i and the period when the *Rank*th refueling transaction was performed.

  For every $i = 0, \ldots, N$, a state E is provided with its current Bellman value W^{Prod}.
  - **Initial state** is $E^{\text{Start}} = (0, H_0, 0, 0)$, with related value $W^{\text{Prod}} = 0$, and time value $i = 0$;
  - **Final states** are states $E^{\text{End}} = (Z, V^{\text{Tank}} \geq H_0, Q, 0)$, associated with a time value $i \leq N$: notice that the process may not be finished when the last refueling transaction takes place.

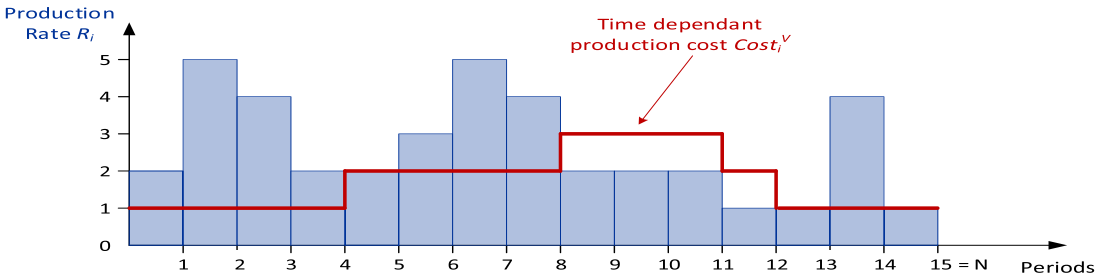

FIGURE 7. Activation and time-dependent production costs for the micro-plant of Figure 3.

– **Decision/Transitions:** for any $i = 0, \ldots, N$, $E = (Z, V^{\text{Tank}}, Rank, Delay)$, a decision is defined as a 2-uple (z, δ) in $\{0,1\}^2$, with the following meaning:
  - $z = 1$ means that the micro-plant will produce during period i;
  - $\delta = 1$ means that the vehicle will perform its $(Rank + 1)$th refueling transaction during period i.

Since production and refueling transactions cannot be performed simultaneously, there are only 3 possible decisions.

- $\underline{z = 1, \delta = 0}$: it requires $(V^{\text{Tank}} + R_i \leq C^{\text{MP}})$; At time $(i+1)$ resulting state E_1 will be:
  - If $Rank \leq Q - 1$ then we should also have $(i \leq M_{Rank+1} - 1)$. We get $E_1 = (1, V^{\text{Tank}} + R_i, Rank, Delay + 1)$, and related transition cost is equal to $\alpha.p + (Cost^F.(1 - Z) + Cost_i^V)$;
  - If $Rank = Q$ then $E_1 = (1, V^{\text{Tank}} + R_i, Rank, 0)$, and related transition cost is equal to $(Cost^F.(1 - Z) + Cost_i^V)$;
- $\underline{z = 0, \delta = 0}$: at time $(i + 1)$ resulting state E_1 will be:
  - If $Rank \leq Q-1$ then we should also have $(i \leq M_{Rank+1}-1)$. We get $E_1 = (0, V^{\text{Tank}}, Rank, Delay+1)$, and related transition cost is equal to $\alpha.p$;
  - If $Rank = Q$ then $E_1 = (0, V^{\text{Tank}}, Rank, 0)$, and related transition cost is equal to 0;
- $\underline{z = 0, \delta = 1}$: it requires $(Rank \leq Q - 1) \wedge (V^{\text{Tank}} \geq \mu_{Rank+1}) \wedge (M_{Rank+1} \geq i \geq m_{Rank+1}) \wedge (Delay \geq B_{Rank})$; Then we get $E_1 = (0, V^{\text{Tank}} - \mu_{Rank+1}, Rank + 1, 1)$, and related transition cost is equal to $\alpha.p$.

Example 5.1. Let us consider the micro-plant of Figure 4, together with the input Example 2.1: $p = 2$, $H_0 = 4$, $T\text{Max} = 30$, $Cost^F = 7$, $C^{\text{MP}} = 15$, $\alpha = 1$, together with the following *Reduced Refueling Strategy* (Fig. 7):

– $Q = 2$;

q	μ_q	m_q	M_q	B_q
0	*	0	3	3
1	14	2	5	8
2	11	10	13	*

Let us suppose that $i = 11$ and that related state is: $Z = 0$; $V^{\text{Tank}} = 12$, $Rank = 1$; $Gap = 9$. Then we see that the following decisions are possible:

– $\underline{z = 1, \delta = 0}$: resulting state is $Z = 1; V^{\text{Tank}} = 13, Rank = 1; Gap = 10$, with transition cost $7 + 2 + 2 = 11$.
– $\underline{z = 0, \delta = 0}$: resulting state is $Z = 0; V^{\text{Tank}} = 12, Rank = 1; Gap = 10$, with transition cost $= 2$.
– $\underline{z = 0, \delta = 1}$: resulting state is $Z = 0; V^{\text{Tank}} = 1, Rank = 2; Gap = 1$, with transition cost $= 2$.

5.2. Filtering devices and greedy Greedy_PM algorithm

As in the case of *DPS_EPC*, we may enhance *DPS_PM* through filtering devices:

- **Filtering through rounding:** as in Section 3.1, we may round values V^{Tank} and W^{Prod} and turn *DPS_MP* into a parametrized algorithm *DPS_PM*(K) which is time-polynomial for any fixed K, and such that, for any value $\varepsilon > 0$, K may be chosen in such a way that in case MP admits an optimal solution with value $W^{\text{Prod-Opt}}$, then *DPS_PM* (K) yields a solution which is feasible with regards to initial value $(1 + \varepsilon/2)$. H_0 and capacity values $(1 + \varepsilon)$. C^{MP}, and whose cost value is no larger than $W^{\text{Prod-Opt}}$.
- **Dominance based filtering rules:** for any i, if *states* $E_1 = (Z_1, V_1^{\text{Tank}}, Rank_1, Delay_1)$ and $E_2 = (Z_2, V_2^{\text{Tank}}, Rank_2, Delay_2)$, given together with values W_1^{Prod} and W_2^{Prod} are such that: $W_1^{\text{Prod}} \leq W_2^{\text{Prod}}$; $Z_2 \leq Z_1$; $(Rank_1 \geq Rank_2) \vee ((Rank_1 = Rank_2) \wedge (Delay_1 \geq Delay_2))$ then *kill* E_2.
- **Logical filtering rule** (check the feasibility of the process with regards to production capacity). For any i, if *state* $E = (Z, V^{\text{Tank}}, Rank, Delay)$ is such that: $(V^{\text{Tank}} + \Sigma_{M_Q > k \geq i} R_k < \Sigma_{Rank+1 \leq q \leq Q} \mu_q)$ or $(V^{Tank} + \Sigma_{k \geq i} R_k < (\Sigma_{Rank+1 \leq q \leq Q} \mu_q + H_0)$ then *kill E*.
- **Quality based filtering rule:** it involves, as in Section 3.2.2, the pre-computed function *Cost_Min*(i, V, Z), which provides us with the minimal cost required from the *micro-plant* to produce V energy units between time $p.i$ and time TMax, Z denoting the state of the micro-plant at the end of period $i - 1$. This function allows us to turn *DPS_PM* into a greedy algorithm *Greedy_PM*:
  - For any i and any related *state* $E = (Z, V^{\text{Tank}}, Rank, Delay)$, hydrogen quantity which remains to be produced is $V = \Sigma_{q \geq Rank+1} \mu_q + H_0 - V^{\text{Tank}}$;
  - By the same way, the Qth refueling transaction cannot take place before period $m_Q + i - Delay - m_{Rank}$;
  - So *Greedy_DPM* works by keeping, for any i, only one state E, and choosing related feasible decision (z, δ) in such a way that resulting state $E_1 = (Z_1, V_1^{\text{Tank}}, Rank_1, Delay_1)$ be consistent with above logical filtering rule and that: $Cost\_Min(i+1, \Sigma_{q \geq Rank_1+1} \mu_s + H_0 - V_1^{\text{Tank}}, Z_1) + \alpha.(m_Q - Delay_1 - m_{Rank_1} +$ (Cost of the transition$(i, E) - > (i + 1, E_1)$) be minimal.

  This greedy algorithm may be applied in order to provide us (in case of success) with an initial production strategy (z, δ) and its value $W^{\text{Prod}}{}_{Init}$. Then our *Quality Based Filtering* rule may be formulated as follows: For any i, if *state* $E = (Z, V^{\text{Tank}}, Rank, Delay)$ and related value W^{Prod} are such that $W^{\text{Prod}} + Cost\_Min\left(i, \Sigma_{q \geq Rank+1} \mu_q + H_0 - V^{\text{Tank}}, Z\right) + \alpha.\,(m_Q - Delay - m_{Rank}) \geq W_{\text{Init}}^{\text{Prod}}$, then *kill E*.

6. Making VD and PM collaborate: The VD->PM pipe-line

The *VD_PM* Decomposition of Section 2.2.3 suggests that the simple way to implement a collaboration between the *Vehicle Driver* and the *Production Manager* is to make them communicate through a one-way pipe-line: Once coefficient β, which determines the objective function of VD and which behaves as the main communication link between VD and PM, has been fixed, we may compute a *Refueling Strategy* (x, L) for the vehicle through the *DPS_VD* algorithm, turn it into a *Reduced Refueling Strategy* (Q, μ, m, M, B) and apply the *DPS_PM* algorithm of Section V.

6.1. Handling EPC through a VD_PM_Pipe-Line algorithm

***VD_PM_Pipe-line* algorithm:**
Input/Output: The same as for the EPC model.
Main Steps:

(1) Choose the scaling coefficient β of the VD model;
(2) Apply *DPS_VD* to resulting VD model: get related *Refueling Strategy* (x, L) and retrieve *Reduced Refueling Strategy* (Q, μ, m, M, B) as in Section 4.3;
(3) Compute $W_{\text{Init}}^{\text{Prod}}$ through *GREEDY_PM* of Section 5.2;
(4) Apply *DPS_PM* augmented with filtering devices of Section 5.2 to the *Reduced Refueling Strategy* (Q, μ, m, M, B): get *Production Strategy* (z, δ);

TABLE 2. Instances.

Num instance	N	M	TMax	p	C^{Veh}	C^{MP}
0	15	4	60	4	24	50
1	78	10	78	1	27	27
2	94	10	94	1	17	34
3	114	10	114	1	23	23
4	99	10	99	1	19	19
5	59	10	118	2	22	22
6	36	10	72	2	19	57
7	78	10	156	2	22	22
8	57	10	114	2	23	23
9	26	8	104	4	12	36
10	26	10	104	4	15	30
11	26	10	96	4	25	25
12	30	10	120	4	17	17
13	33	8	132	4	26	26
14	20	8	80	4	15	30
15	27	8	108	4	26	78
16	50	10	200	4	28	28
17	24	10	96	4	24	72
18	44	10	176	4	20	40
19	32	12	128	4	28	28
20	32	12	128	4	28	28
21	45	14	180	4	24	24
22	30	8	120	4	29	29
23	26	8	104	4	22	66
24	16	10	68	4	14	42
25	19	10	76	4	16	48
26	20	10	80	4	27	27
27	50	12	200	4	18	36
28	50	12	200	4	18	36

(5) **Retrieve** activation vector $y = (y_i, i = 0, \ldots, N-1)$, time vector $T = (T_j, j = 0, \ldots, M+1)$, load vector $L^* = (L_i^*, i = 0, \ldots, N-1)$, and global cost $\Sigma_{i=0,\ldots,N-1}(Cost^F.y_i + Cost^V_{i.}.z_i) + \alpha.T_{M+1}$.

6.2. Discussion

As we told at the beginning of the paper, our purpose here is to implement synchronization while emulating natural interaction processes which are likely to take place between decentralized players. The one-way pipeline mechanism is clearly one of the simplest ones. Of course, one could think in designing more sophisticated interaction schemes. In any case, some points may be discussed:

– **About the mere structure of the VD_PM algorithm:** the VD_PM link defined by β behaves here like a mono-directional link: One assigns a value to β, and successively runs *DPS_VD* and *DPS_PM*, while possibly trying several shots. But one may ask about retrieving information from this $DPS\_VD -> DPS\_PM$ sequence and adapting β accordingly. For instance, one may try to implement a fixed-point loop, and, after any run of the $DPS\_VD -> DPS\_PM$ sequence, assign β a new value β^* such that $\beta^*.(\Sigma_j L_j.x_j)$ quantity becomes equal to the energy cost $\Sigma_{i=0,\ldots,N-1}(Cost^F.y_i + Cost^V_{i.}.z_i)$ and start again. We did not try this option, mainly because it is time-consuming. By the same way, one could make parameter α, which weights the waiting times that the *Vehicle Driver* has to accept, be part of the negotiation process.

TABLE 3. Values *Exact_ DPS_EPC*, *Pipe_Line* and *Greedy_EPC*.

Instance	*Pipe_Line*	*Greedy_50*	*Exact_DPS_EPC*
0	47	46	46
1	97	103	97
2	129	141	129
3	140	144	131
4	157	163	157
5	113	113	109
6	103	97	97
7	140	176	140
8	150	150	141
9	116	116	116
10	133	133	133
11	187	−1	184
12	243	248	232
13	182	201	182
14	83	81	81
15	104	100	100
16	133	139	131
17	102	102	102
18	126	126	126
19	304	305	297
20	304	305	297
21	325	309	305
22	199	199	199
23	141	141	141
24	65	65	65
25	107	107	107
26	124	121	121
27	209	202	202
28	209	202	202

– **About the choice of β:** β should reflect the production cost of the energy which has to be produced. But, since we do not *a priori* know the temporal distribution of the refueling transactions, we split the time interval into 3 macro-periods and do as if this distribution were to be uniform between those macro-periods. So we set:
  - H = Energy that the vehicle has to load according to the VD taken with $\beta = 1$ and $\alpha = 0$;
  - $Rough\_Cost = Cost\_Min(0,\ H/3,\ 0) + Cost\_Min(\lfloor N/3 \rfloor,\ H/3,\ 0) + Cost\_Min(2\lfloor N/3 \rfloor,\ H/3,\ 0)$;
  - $\beta = Rough\_Cost/H$.
– **About the specification of the VD model:** we may notice that it would be possible to deal with VD the same way as we did, while replacing the term $\beta.(\Sigma_j L_j.x_j)$ of the objective function by a term $(\Sigma_j \beta_j.L_j.x_j)$, which would give us more flexibility. Still, anticipating the price of hydrogen amount loaded by the vehicle at station would remain a difficulty.

7. NUMERICAL EXPERIMENTS

Purpose: What we want here is to evaluate:

TABLE 4. Maximal states per time units generated through *DPS_VD*, *DPS_PM* and *Greedy_EPC*.

Instance	*State_VD*	*State_PM*	*State_DPS_EPC*
0	1	49	421
1	15	3024	31 810
2	11	4187	22 394
3	19	4760	40 032
4	1	4161	32 765
5	1	2626	53 053
6	1	2900	28 876
7	1	4404	81 790
8	19	1647	16 028
9	1	1730	13 476
10	11	1409	8060
11	1	312	4664
12	1	530	11 368
13	1	770	18 761
14	1	872	8078
15	19	2193	352
16	16	3222	118 795
17	11	2022	31 415
18	11	5320	26 639
19	1	283	11 543
20	1	283	11 543
21	1	1725	28 073
22	1	573	13 544
23	5	1159	10 989
24	5	629	6013
25	1	174	887
26	19	76	1718
27	1	6497	41 858
28	1	6497	41 858

(1) The pipe-line scheme *DPS-VD* $->$ *DPS-PM*: we focus on the tradeoff induced by this scheme between accuracy (obtained through exact *DPS-EPC* algorithm) and running cost (summarized by the number of states visited throughout the process).
(2) The filtering power of the devices for the general exact *DPS-EPC* algorithm which were described in Section 3.2. We focus here on the number of states visited during the process, depending on the filtering devices which are activated.

Instances: we fix N and M, and randomly generate stations j and *Depot* and the *Micro-Plant* as point of the 2-dimensionnal Euclidian space. Then d_j, d_j^* and t_j, e_j, ε_j, ε_j^* respectively corresponds to Euclidean distance and Manhattan distance, rounded in such a way they take integral values and remain consistent with triangle inequlities. Then we fix C^{MP}, C^{Veh} and $T\mathrm{Max}$, in such a way it ensures the existence of a feasible solution. Finally, we fix the cost coefficients, in such a way that the fixed cost $Cost^F$ is at least equal to the largest coefficient $Cost_i^V$, $i = 0, \ldots, N-1$. In the present case, we use a set of 29 instances, with characteristics *N, M,* $T\mathrm{Max}$*, p,* C^{Veh}, C^{MP} as follows:

Technical context: algorithms were implemented in C++, on a computer running Windows 10 Operating system with an IntelCore i5-6500@3.20 GHz CPU, 16 Go RAM and Visual Studio 2017 compiler.

Outputs 1: in order evaluate the pipe-line scheme *DPS_VD* $->$ *DPS_PM*, we first run, for any instance, a multi-start version of (randomized) *Greedy_EPC* with 50 replications, and get a value *Greedy_50*. Next we

TABLE 5. Related CPU times (in seconds).

Instance	*CPU_Pipe-Line (s)*	*CPU_Greedy_EPC-50 (s)*	*CPU_DPS_EPC (s)*
0	0.0068	0.00035	0.033633333
1	0.6456	0.0027	25.54665
2	1.31	0.00215	11.94846667
3	1.2269	0.001716667	39.99966667
4	1.073583333	0.00045	58.08111667
5	0.43275	0.000266667	39.82303333
6	0.272583333	0.000766667	10.80141667
7	0.956283333	0.0004	104.1274167
8	0.190116667	0.001116667	11.85403333
9	0.1447	0.0014	1.170633333
10	0.079866667	0.0007	1.2939
11	0.01795	6.66667E-05	0.818283333
12	0.042216667	0.0001	1.943283333
13	0.047783333	0.000116667	4.701516667
14	0.039616667	6.66667E-05	0.834816667
15	0.147916667	3.33333E-05	0.283533333
16	0.434816667	0.0008	54.20976667
17	0.097083333	0.001383333	3.127866667
18	0.61875	0.000983333	22.05958333
19	0.0313	0.000116667	1.596916667
20	0.03065	0.00005	1.605516667
21	0.222483333	0.000283333	15.45353333
22	0.0476	8.33333E-05	2.958133333
23	0.080016667	0.000733333	1.029416667
24	0.028266667	0.00005	0.28065
25	0.00805	3.33333E-05	0.027033333
26	0.002816667	3.33333E-05	0.02595
27	0.904333333	0.001033333	31.98286667
28	1.0297	0.001066667	31.89528333

run *VD_PM_Pipe_Line* and get a value denoted by *Pipe_Line*. Finally we run *DPS_EPC* and get exact optimal *Exact_DPS_EPC* Value. So we provide the following outputs:

- **Table 2:** values *Pipe_Line*, *Greedy_50* and *Exact_DPS_EPC*.
- **Table 3:** maximal number of states *State_VD*, *State_PM* and *State-Exact_DPS_EPC* at any time unit, for the DPS algorithms *DPS_VD*, *DPS_PM* and *DPS_EPC*.
- **Table 4:** CPU Times *CPU_Pipe-Line*, *CPU_Greedy_EPC_50* and *CPU_DPS_EPC* for the DPS algorithms *VD_-PM_Pipe-Line*, *Greedy_EPC_50* and *DPS_EPC*.

Comments: the pipe-line scheme *DPS_Vehicle* $->$ *DPS_Production* involves significantly less states and CPU time, for a gap almost negligible.

Outputs 2: we focus on the filtering power of the various devices introduced in Section 3.2.2. Once again we run exact *DPS_EPC*, while focusing on the maximal number of states per time units. For any instance, we provide (Tab. 5):

(1) *ST*_Max = Maximal number of states for a given pair (i, j) obtained when only considering the strong dominance rule.
(2) *ST_LOG* = Maximal number of states obtained for a given (i, j) when only applying the *Logical Filtering.*

TABLE 6. Filtering power of logical and quality based filtering rules.

Instance	*ST_Max*	*ST_LOG*	*State_DPS_EPC*
0	576 000	421	421
1	1 137 240	31 810	31 810
2	1 086 640	22 394	22 394
3	1 206 120	40 032	40 032
4	714 780	32 765	32 765
5	1 142 240	53 053	53 053
6	1 559 520	28 876	28 876
7	1 510 080	81 790	81 790
8	1 206 120	16 028	16 028
9	718 848	13 476	13 476
10	936 000	8060	8060
11	300 000	4800	4664
12	173 400	14 929	11 368
13	356 928	18 761	18 761
14	576 000	8078	8078
15	876 096	48 501	352
16	3 136 000	118 795	118 795
17	3 317 760	31 415	31 415
18	2 816 000	26 639	26 639
19	602 112	11 589	11 543
20	602 112	11 589	11 543
21	2 903 040	28 073	28 073
22	1 614 720	13 608	13 544
23	2 416 128	10 989	10 989
24	199 920	6013	6013
25	291 840	887	887
26	291 600	1718	1718
27	3 110 400	41 858	41 858
28	3 110 400	41 858	41 858

(3) *State_DPS_EPC* = Maximal number of states obtained for a given (i, j) when applying all filtering rules.

Comments: one sees that, while the filtering rules based on *Strong Dominance* have little filtering impact, those based upon logical anticipation and optimistic estimation are significantly more efficient. Still, we keep handing a large amount of states as soon as M and N become large. The pipe-line scheme *DPS_Vehicle* $->$ *DPS_Production* involves significantly less states and CPU time, for a gap almost negligible (Tab. 6).

8. CONCLUSION

We have been presenting here a collaborative dynamic programming scheme in order to solve a scheduling problem which requires synchronizing mechanisms. Many issues remain to be addressed: Extending our approach to several vehicles; dealing with uncertainties related to hydrogen production; casting the routing issue into the decision process; adapting the algorithms to *one line* or dynamic decision making.

Acknowledgements. We are thankful to LABEX IMOBS3 and French ANR, as well as to PGMO: Gaspard Monge Program for Optimization, for supporting and funding this research.

References

[1] S. Albers, Energy-efficient algorithms. *Commun. ACM* **53** (2010) 86–96.

[2] E. Angel, E. Bampis and V. Chau, Low complexity scheduling algorithms minimizing the energy for tasks with agreeable deadlines. *Discrete Appl. Math.* **175** (2014) 1–10.

[3] P. Baptiste, Scheduing unit tasks to minimize the number of idle periods: a polynomial time algorithm for offline dynamic power management. In: Proc. 17 th Annual ACM-SIMA Symposium on Discrete Algorithms (2006) 364–367.

[4] P. Baptiste, E. Neron and F. Sourd, Modèles et Algorithmes en Ordonnancement. Ed Ellipses (2004) 198–203.

[5] L. Benini, A. Bogliolo and G. De Micheli, A survey of design techniques for system level dynamic power management. *IEEE Trans. Very Large Scale Integr. Syst.* **8** (2000) 299–316.

[6] K. Biel and C.H. Glock, Systematic literature review of decision support models for energy efficient decision planning. *Comput. Ind. Eng.* **101** (2016) 243–259.

[7] A. Burke, Batteries and ultracapacitors for electric, hybrid, and fuel cell vehicles. *Proc. IEEE* **95** (2007) 806–820.

[8] C.C. Chan, The state of the art of electric, hybrid, and fuel cell vehicles. *Proc. IEEE* **95** (2007) 704–718.

[9] P. Chretienne and A. Quilliot, Homogeneous non idling schedules of unit-time jobs on identical parallel machines. *Discrete Appl. Math.* **161** (2013) 1586–1597.

[10] P. Chretienne and A. Quilliot, A polynomial algorithm for the homogeneous non idling scheduling problem of unit-time independent jobs on identical parallel machines. *Discrete Appl. Math.* **243** (2018) 132–139.

[11] P. Chretienne, P. Fouilhoux and A. Quilliot, Anchored reactive and proactive solutions to the CPM-scheduling problem. *Eur. J. Oper. Res.* **261** (2017) 67–74.

[12] E. Demaine, M. Ghodsi and M. Taghi Hajiaghayi, Scheduling to minimize gaps and power consumption. *SPAA* (2007) 46–54.

[13] M. Drexl, Synchronization in vehicle routing: a survey of VRPs with multiple synchronization constraints. *Transp. Sci.* **46** (2012) 297–316.

[14] J.R. Duflou, J.W. Sutherland, D. Dornfeld, C. Herrmann, J. Jeswiet, S. Kara, M. Hauschild and K. Kellens, Toward energy and resource efficient manufacturing: a process and system approach. *CIRP Ann. – Manuf. Technol.* **61** (2012) 587–609.

[15] S. Erdogan and E. Miller-Hooks, A green vehicle routing problem. *Transp. Res. Part E Logistics Transp. Rev.* **109** (2012) 100–114.

[16] A. Franceschetti, E. Demir, D. Honhon, T. Van Woensel, G. Laporte and M. Stobbe, A metaheuristic for the time dependent pollution-routing problem. *Eur. J. Oper. Res.* **259** (2017) 972–991.

[17] A. Giret, D. Trentesaux and V. Prabhu, Sustainability in manufacturing operations scheduling: a state of art review. *J. Manuf. Syst.* **37** (2015) 126–140.

[18] C. Grimes, O. Varghese and S. Ranjan, Light, Water, Hydrogen: The Solargeneration of Hydrogen By Water Photoelectrolysis. Springer-Verlag US (2008).

[19] S. Irani and K. Pruhs, Algorithmic problems in power management. *SIGACT News* **36** (2003) 63–76.

[20] I. Kara, B.Y. Kara and M. Kadri Yetis, Energy minimizing vehicle routing problem, edited by A. Dress, Y. Xu and B. Zhu. In: Combinatorial Optimization and Applications. Berlin, Heidelberg (2007) 62–71.

[21] C. Koç, O. Jabali, J. Mendoza and G. Laporte, The electric vehicle routing problem with shared charging stations. In: *Int. Trans. Oper. Res.* **26** (2018) 1211–1243.

[22] Y. Kuo, Using simulated annealing to minimize fuel consumption for the time-dependent vehicle routing problem. *Comput. Ind. Eng.* **59** (2010) 157–165.

[23] A. Lajunen, Energy consumption and cost-benefit analysis of hybrid and electric city buses. *Transp. Res. Part C: Emerg. Technol.* **38** (2014) 1–15.

[24] D. Lamy, *Méthodes et outils pour l'ordonnancement d'ateliers avec prise en compte de contraintes additionnelles : énergétiques et environnementales*. Thèse de Doctorat, Université Clermont-Auvergne. France (2017).

[25] S. Licht, Thermochemical and Thermal/Photo Hybrid Solar Water Splitting. Springer New York, New York, NY (2008).

[26] C. Lin, K.L. Choy, G.T. Ho, S.H. Chung and H. Lam, Survey of green vehicle routing problem: past and future trends. *Expert Syst. App.* **41** (2014) 1118–1138.

[27] F.J. Mailfert, E. Mole Kamga, A. Quilliot and H. Toussaint, Simultaneous Management of Energy Production and Consumption. In: Proc. IEEE CODIT 2020 (2020) 8.

[28] G. May, I. Barletta, B. Stahl and M. Taisch, Energy management in production: A novel method to develop key performance indicators for improving energy efficiency. *Appl. Energy* **149** (2015) 46–61.

[29] J.-Y. Moon and J.-W. Park, Smart production scheduling with time-dependent and machine-dependent electricity cost by considering distributed energy resources and energy storage. *IJPR* **52** (2013) 3922–3939.

[30] J.-Y. Moon, K. Shin and J.W. Park, Optimization of production scheduling with time-dependent and machine-dependent electricity cost for industrial energy efficiency. *Int. J. Adv. Manuf. Technol.* **68** (2013) 523–535.

[31] H. Mustapha, C. Desdouits, R. Giroudeau, C. Le Pape, Production scheduling with a piecewise-linear energy cost function. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI). (2016) 1–8.

[32] A. Pechmann and I. Schöler, Optimizing energy costs by intelligent production scheduling, edited by J. Hesselbach and C. Herrmann. In: Glocalized Solutions for Sustainability in Manufacturing. Berlin, Heidelberg (2011) 293–298.

[33] M. Raylan, S. Matos, Y. Frota and L. Satoru Ochi, Green vehicle routing and scheduling problem with split delivery. Joint, EURO/ALIO, International Conference 2018 on Applied Combinatorial Optimization, (EURO/ALIO 2018). *Electron. Notes Discrete Math.* **69** (2018) 13–20.

[34] M. Sachenbacher, M. Leucker, A. Artmeier and J. Haselmayr, In: Efficient energy-optimal routing for electric vehicles. AAAI (2011).

[35] M. Schneider, A. Stenger and D. Goeke, The electric vehicle-routing problem with time windows and recharging stations. *Transp. Sci.* **48** (2014) 500–520.

[36] R. Waraich, M. Galus, C. Dobler, M. Balmer, G. Andersson and K. Axhausen, Plug-in hybrid electric vehicles and smart grid: micro simulation. *Transp. Res. C* **28** (2014) 74–86.