Synchronizing energy production and vehicle routing

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

▶ **To cite this version:**

Fatiha Bendali, Eloise Mole Kamga, Jean Mailfert, Alain Quilliot, Hélène Toussaint. Synchronizing energy production and vehicle routing. RAIRO - Operations Research, 2021, 55 (4), pp.2141-2163. 10.1051/ro/2021093 . hal-03282242

HAL Id: hal-03282242
https://hal.science/hal-03282242v1

Submitted on 8 Jul 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

RAIRO-Oper. Res. 55 (2021) 2141–2163
https://doi.org/10.1051/ro/2021093

RAIRO Operations Research
www.rairo-ro.org

SYNCHRONIZING ENERGY PRODUCTION AND VEHICLE ROUTING

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

Abstract. The emergence of locally produced renewable energies induces the appearance of a new generation of local energy players, which are at the same time producers and consumers. In case of time dependent solar energy production, it raises the question of synchronizing production and consumption. We deal here with this issue, in the context of an experimental Solar Hydrogen (H_2) production platform. More precisely, we try here to simultaneously schedule a H_2 fueled vehicle which follows a pre-computed route while being compelled to periodically refuel, and the H_2 production micro-plant which is required to produce related energy under time dependent production costs and productivity rates, both processes being subject to storage capacity constraints. In order to do it, we design a global dynamic programming (DP) algorithm for the resulting NP-Hard problem. This DP algorithm involves a 2D time space which links energy consumption by the vehicle and its production by the micro-plant. Since the number of states induced by this DP algorithm becomes an issue as soon as the size of the problem increases, we first propose a theoretical Polynomial Time Approximation Scheme (PTAS), next design several practical pruning devices and finally perform numerical tests in order to check their efficiency.

Mathematics Subject Classification. 90-10, 90C39, 90-08, 90C05.

Received March 4, 2020. Accepted June 17, 2021.

1. Introduction

The rise of renewable energy sources (photovoltaic, wind, hydrogen or biomass based: see [6, 7, 15, 21]), which aim at replacing CO_2 emissions by clean electrical power, together with the current scarcity of related charge/recharge infrastructures, have been motivating during the last decades an interest from O.R. researchers to *green* issues.

Most contributions were related to electric or hybrid vehicles where the goal is to minimize energy consumption (*Green VRP*, *Pollution-Routing Problem* and *Hybrid Vehicle Problem*: see [13, 14, 28]). Related models consider recharge transactions submitted to time windows or shared access constraints and aim at minimizing some mixed cost which partially reflects environmental concerns. *Green VRP* is for instance introduced in [13] by Erdogan *et al.* with the purpose of promoting either the minimization of fuel consumption or the use of alternative-fuel powered vehicles. Their model is an extension of standard VRP involving limited fuel tank capacities and an objective function which mixes total travel distance and the number of refuelling

Keywords. Scheduling, dynamic programming, energy.

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

© The authors. Published by EDP Sciences, ROADEF, SMAI 2021

This is an Open Access article distributed under the terms of the Creative Commons Attribution License (https://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

transactions. Numerical handling relies on a MILP: *Mixed Integer Linear Programming* formulation and local search heuristics. In [14], Franceschetti *et al.* tackle the *pollution-routing* problem which aims at minimizing a cost that depends on driver's wages and fuel consumption while considering time-dependent traffic congestion. They use an ALNS: *Adaptative Large Neighbourhood Search* metaheuristic in order to handle a model whose main complexity lies in the time dependency of the costs of the arcs which define the transit network. In [28], Raylan *et al.* address what they call *Green Vehicle Routing*, which means the problem of routing a fleet of vehicle inside a network whose arcs are provided with time dependent lengths, while minimizing a cost based on CO_2 emission due to both the routes and the vehicle loads. They manage resulting model through multi-start ILS. In [17], Kara *et al.* follow a similar approach and propose a variant of the well-known CVRP: *Capacitated Vehicle Routing Problem* by setting a new cost function, which also involves the loads of the vehicles and their impact on energy spending. They propose an ILP to solve their model. In [29], Sachenbacher *et al.* also deal with time dependency, but they introduce specific recharging schemes and consequently adapt shortest path computation through an adaptation of A* algorithm. In [19], Kuo addresses time dependency through the prism of energy consumption and sets a routing model involving time windows, where the drivers may reach an optimal tradeoff between energy consumption, transportation distance and transportation time by conveniently controlling speed. He designs a simulated annealing algorithm which computes routing strategies with lower fuel consumption but longer transportation times and transportation distances. In [30], Schneider *et al.* apply a hybrid VNS&Tabu scheme to the handling of a VRP model with time windows, which includes specific battery recharging schemes. In [22], Lin *et al.* provide a survey on the *Green Vehicle Routing* problem and the *Pollution Routing* problem, which both make speed control and traffic jam avoidance part of the models.

In [18], Koç *et al.* deal with electric vehicle routing problems which refers not only to the assignment of the customers to the vehicles and their sequencing, but also to where and how much the vehicles are recharged: The feasibility of the routes is therefore constrained by the characteristics of recharging configurations (kind of charging functions: linear, piecewise linear, ...), and the autonomy of the vehicles. This last issue is also addressed in our paper. In [18] authors consider customer sequencing as part of the problem, and study an *Electric*-VRP model with non-linear charging functions, multiple charging technologies and variable charging quantities, while explicitly accounting for the number of chargers available at privately managed charging stations. They handle their model first through a MILP formulation, and next by through a metaheuristic scheme which alternatively generates routes and schedules the refueling transactions along those routes.

Still, not all contributions put the focus on optimization: In [31], Waraich *et al.* propose a multi-agent discrete event simulation model in order to evaluate the impact of the time dependence of both demands and costs on the behavior of a fleet of *last mile* electric vehicles. In [20], Lajunen uses a vehicle simulation method in order to compare the reductions of energy consumption and emissions induced by different configurations of urban shuttle&bus fleets. They take into account capital costs, operating costs and costs induced by energy storage devices.

While transportation has been paid most attention by O.R. communities, some authors have been addressing the issue of scheduling an industrial process (see [4, 11]) while taking into account temporal variations of the energy costs, access restrictions and environmental concerns. In [24], Moon *et al.* Deal with parallel machine scheduling while taking into account time dependent energy costs, and in [25], they address flexible Job Shop with the same kind of hypothesis. In both case, they use MILP formulation and genetic algorithms. In [26], Mustapha *et al.* schedule an industrial production process subject to piecewise linear electric cost while using MILP models and dynamic programming algorithms. In [27], Pechman and Schöler focus on the management of energy consumption processes which involve peaks and breaks, as in steel industry or inside large buildings. In [16], Irani and Pruhs address the specific case of information processing and propose a survey about existing scheduling models and algorithms for the minimization of energy consumption inside a data center or a cloud architecture. In [1], Albers deals with the same issue, and provides a survey about algorithmic scheduling and resource allocation techniques, mostly based on heuristic decision rules designed for dynamic contexts, for reducing energy consumption of current computers, including techniques like sleep states and power-down mechanisms, dynamic speed scaling and temperature management.

While most articles have been related to applications, we should mention the existence of several theoretic studies which deal with complexity and approximation issues, for models which put at stake the cost of idleness and the impact of time dependencies (see [1, 8, 16]). Typically, in [2, 3, 11], respectively Angel *et al.*, Baptiste and Demaine *et al.*, rely on dynamic programming algorithms in order to explore models involving energy costs induced by set up transactions and gaps in the use of resources, which may be solved in pseudo-polynomial time. By the same way, in [9, 10], Chretienne *et al.* propose polynomial time algorithms for specific scheduling models (oven management for the tyre industry, ...) whose costs are mainly due to breaks inside energy consumption processes. On another side, main energy producers have been for a long time carrying on systematic studies about big grain energy production planning (gas, electricity, dam or nuclear plant management), with the purpose of meeting large scale uncertain aggregated demands (see [24, 25]). In [5], Benini *et al.* provide a survey about dynamic power management (DPM), while focusing on multi-level systems which require synchronizing coarse grain production with distributed fine grain distribution.

A current trend in Energy Economics is towards the decentralization of Energy production. This is due to both the deregulation of energy markets and to the rise of technologies, which more and more allow local players to become *small* energy producers (solar, wind, hydrogen, ...), while simultaneously remaining consumers. Those local producer/consumers may be factories, farms and even individual householders. This raises complex questions to traditional energy producers, who lost their monopolistic position but remain key players, because of their control on backbone networks and main hydraulic and nuclear plants. In the context of the activities of Labex IMOBS3 in Clermont-Ferrand, France, devoted to *Innovative Mobility Technologies and Services*, we are participating into a project about the design and control of a local micro-plant for *Solar Hydrogen* (H_2) production. This micro-plant is asked to provide autonomous vehicles with electrical power obtained from hydrogen fuel cells. While most H_2 production is usually performed through power costly electrolysis processes, we assume here that it relies on solar power and photolysis (see [7, 15]), which involve a marginal amount of external electric power but also make the productivity of the process deeply dependent on the intensity of solar illumination. In [21], one may find a review about both existing and prospective ways to generate hydrogen from sunlight and water. While *Solar Hydrogen* currently remains an uncertain goal, at least from a cost perspective, the scientific principles behind its generation are well understood: *Solar Hydrogen* is the largest renewable carbon-free resource among the other renewable energy options, and it may be produced in an almost fully autonomous way.

According to this paradigm, energy production/consumption becomes endogenous, and performed according to a kind of closed loop. It induces a need for high level synchronization between both heterogeneous production and consumption processes. Still, very few works address the issue of synchronizing endogenous transportation and energy production processes (see for instance the survey of Drexl [12], in order to get a review of the contributions related to synchronization in the case of vehicle routing alone). It comes that present work is about the simultaneous management of, on one side, a fleet of small electric vehicles provided with H_2 power cells, which are required to perform local logistic tasks inside a restricted area, and, on the other side, a micro-plant in charge of producing the H_2 fuel which is going to be periodically loaded into those vehicles. Taken as a whole, this management involves forecasting, safety (related to vehicle autonomy) ensuring and synchronizing: one must match the *pickup and delivery* activity of the vehicles with the H_2 production/stock strategy of the micro-plant. Nevertheless, we only address a very specific part of this problem: we consider only one vehicle, which is required to perform tasks according to a pre-fixed order. This vehicle starts its route with some H_2 fuel load, and its tank has a limited capacity. Therefore, it must as in [18] periodically go back to the micro-plant in order to refuel. The micro-plant has a limited production/storage capacity, which depends on solar illumination. Our goal is to simultaneously schedule both the refueling transactions of the vehicle and the production/storage activity of the micro-plant, while minimizing production economical cost and the duration of the vehicle tours.

As mentioned in the title of the paper, we focus here on *synchronization*, and neither deal with uncertainty nor with the design of the vehicle route. Still, since this very generic word *synchronization* may refer to very distinct contexts, among them real time contexts where agents involved into the process are provided with some level of autonomy, we must explain what we mean here: We do not deal here with real time synchronization of both

players (vehicle and micro-plant) and on the various protocols required in order to ensure those players to meet when necessary, even in the case of unforeseen events. Our point of view here remains a standard Combinatorial Optimization one, and suppose the existence of a central manager in charge of *a priori* scheduling the activity of both the vehicle and the micro-plant. It comes that *Synchronization* refers here to the design of a centralized static scheduling algorithm and to the way we link together decisions related to the vehicle with those related to the micro-plant in order to ensure that refueling transactions will be performed in a way consistent with production, consumption and storage constraints.

So the first contribution of the paper consists of a static model related to resulting collaborative scheduling problem. We check that this model is NP-Hard, and propose a dynamic programming scheme (DPS), close to the schemes designed in [3, 11] for energy consumption planning models involving *break* and *set up* costs. *Synchronization* is contained here into the specific structure of this DPS, which involves compound time and state sets in order to link together the vehicle and micro-plant processes.

Then this DPS allows us to state a PTAS (*Polynomial Time Approximation Scheme*) result. Still, in practice, this theoretical result does not keep it from generating to many states as soon as the size of the model increases. So we introduce pruning devices, close to the ones which were introduced in [23] by Lozano and Medeglia in their *Pulse* algorithm for the *Constrained Shortest Path* problem. They are either logical based devices, which aim at anticipating inconsistencies, or upper bound based, involving the pre-computation of a lower bound together with a feasible initial solution. Part of the study is devoted to an experimental evaluation of the filtering power of those devices.

The paper is organized as follows: Section 2 provides the *Synchronous Management of Energy Production and Consumption* (SMEPC) model, together with an ILP formulation. Section 3 first checks that the SMEPC model is NP-Hard, next proposes a global DPS for this model, and finally states a *Polynomial Time Approximation Scheme* (PTAS) like result. Section 4 introduces the filtering devices: logical ones, lower/upper bound based filtering devices and heuristic ones. Section 5 is devoted to numerical experiments, which mainly aim at evaluating the impact of the various filtering devices and the quality of the upper bound heuristic.

2. Synchronized Management of Energy Production and Consumption (SMEPC)

2.1. The model

We consider here some vehicle which has to perform internal logistics tasks, while following a route Γ which starts from some *Depot* node and ends in the same way 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 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 order to perform local tasks. The vehicle may leave *Depot* at time 0 and should finish its route no later than some threshold time $T\text{Max}$.

It happens now that our vehicle is powered by hydrogen (H_2) fuel. The capacity of its tank is denoted by C^{Veh} and we know, for any $j = 0, \ldots, M$, the H_2 amount e_j required in order to move from station j to station $j+1$. The initial H_2 load of the vehicle is denoted by E_0, and the vehicle is required to end its trip with at least the same energy load E_0. It comes that the vehicle will have to 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^* and t_j, as well as quantities e_j, ε_j, ε_j^* are non null and satisfy the *Triangle Inequality* (see Fig. 1).

Figure 2 displays an example of 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. The capacity C^{Veh} of the vehicle is supposed to be equal to 15 and its initial load E_0 is supposed to be equal to 8.

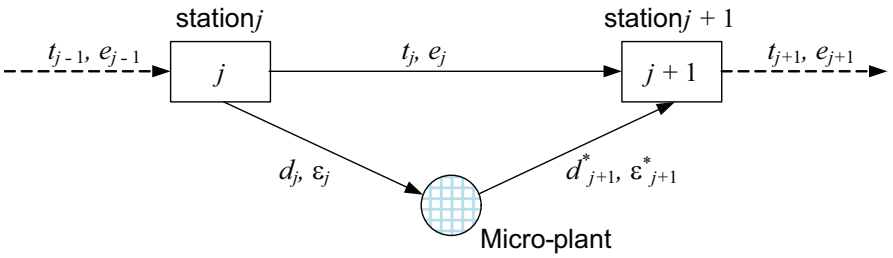

FIGURE 1. Notations regarding time and energy values for 2 consecutive stations j and $j+1 (j = 1, \ldots, M)$.

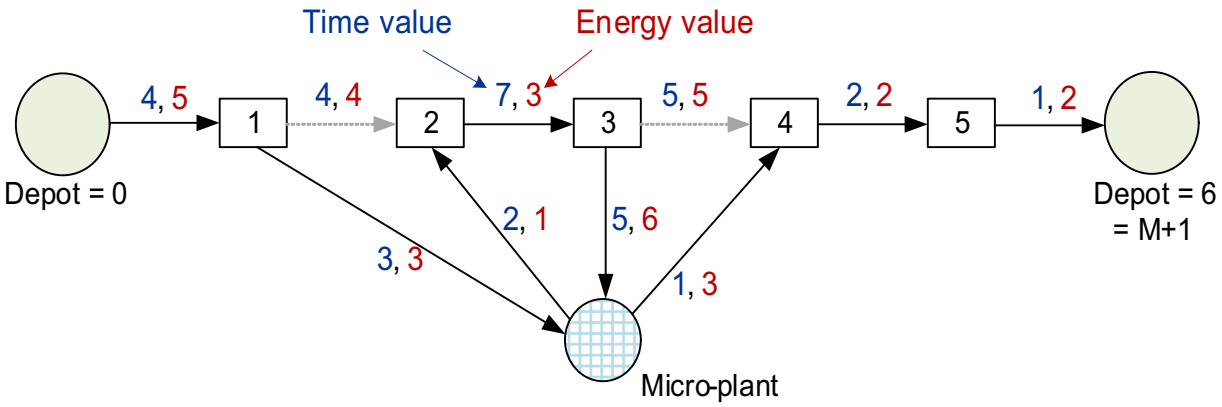

FIGURE 2. A vehicle trip, with its refueling transactions with $M = 5$.

On another side, the micro-plant produces H_2 *in situ* from water through a combination of photolysis and electrolysis. Resulting H_2 is stored inside a tank located directly in the micro-plant, whose capacity (in energy units) is denoted by C^{MP}. We suppose that:

- The time space $\{0, \ldots, T\mathrm{Max}\}$ is divided into periods $P_i = [p.i, p.(i+1)], i = 0, \ldots, N-1$, all with a same length equal to p such that $T\mathrm{Max} = N.p$. For the sake of simplicity, 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, with *production rate* R_i depending on period i. At time 0, the current load of the micro-plant tank is equal to $H_0 \leq C^{\mathrm{MP}}$ and the micro-plant is not active. We impose that the same situation holds at time $T\mathrm{Max}$.
  Figure 3 displays an example of production performed by the micro-plant: periods that are underlined in red corresponds to the periods when the micro-plant is activated. Periods in blue corresponds to the periods when the micro-plant is active.
- Because of safety concerns, the vehicle cannot refuel while the micro-plant is producing. Any vehicle refueling transaction should start at the beginning of some period $i = 0, \ldots, N-1$, and end at the end of period i. Since vehicle refueling and energy production are mutually exclusive, the vehicle may wait at the micro-plant before being allowed to refuel.
- Producing H_2 fuel has a cost, which may be decomposed into 2 components:
  - An *activation cost*, constant and denoted by $Cost^F$, which is charged every time the micro-plant is activated. This activation cost is due to the fact that ensuring the safety of activation requires some human intervention.

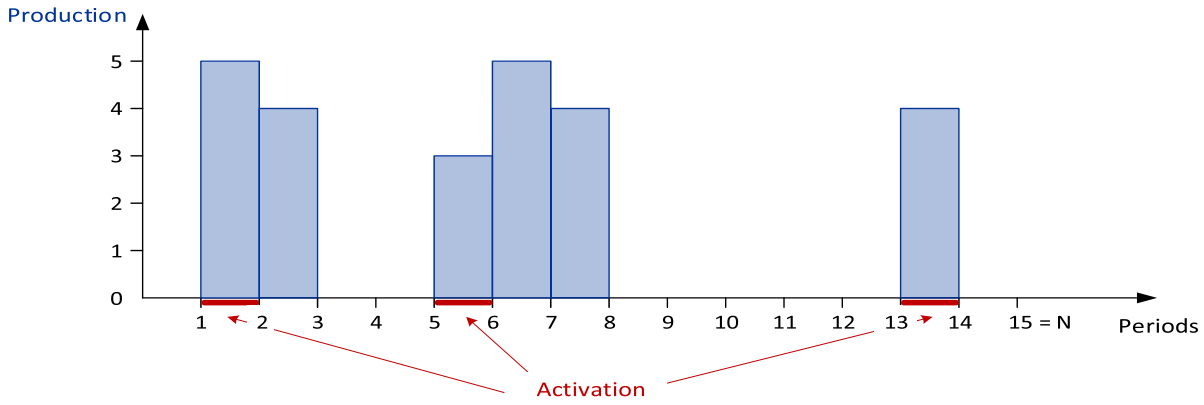

FIGURE 3. An example of micro-plant activity, with $N = 15$.

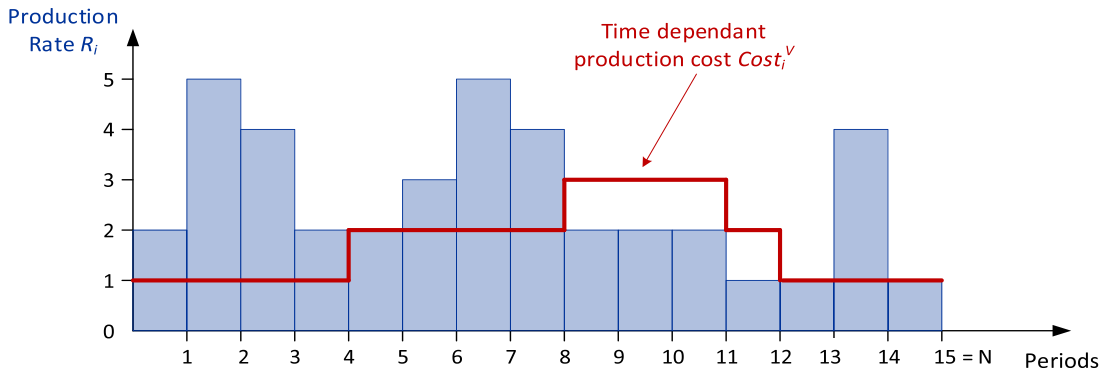

FIGURE 4. Production rates and time-dependent production costs for the micro-plant of Figure 3.

- A *time-dependent* production cost $Cost_i^V$, which corresponds to the power consumed during period i, provided that the micro-plant is active during this period: $Cost_i^V$ is independent on the amount of hydrogen really produced during period i and is only related to the cost of the electrolysis process which, combined to some photolysis process, will derive hydrogen from water. The electricity amount required by this electrolysis process is independent on the amount of hydrogen effectively produced, which depends on the efficiency of the photolysis process (the lightning). It comes that $Cost_i^V$ only reflects the time-indexed prices charged by the electricity provider at period i.

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

Then our *Synchronized Management of Energy Production and Consumption* (SMEPC) 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 H_2 fuel needed by the vehicle.
– Both induced H_2 production cost *Cost* and time T are the smallest possible. We merge both above criteria into a unique one: $Cost + \alpha.T$, where α is a conversion factor from time into economical cost.

Figure 5 below shows the synchronization between the vehicle and the micro-plant of Figures 1–3, which allows us to get a feasible solution in case $p = 2$, $E_0 = 8$, $H_0 = 4$, $T\text{Max} = 30$, $Cost^F = 7$, $C^{\text{MP}} = 15$, $C^{\text{Veh}} = 15$, $\alpha = 1$.

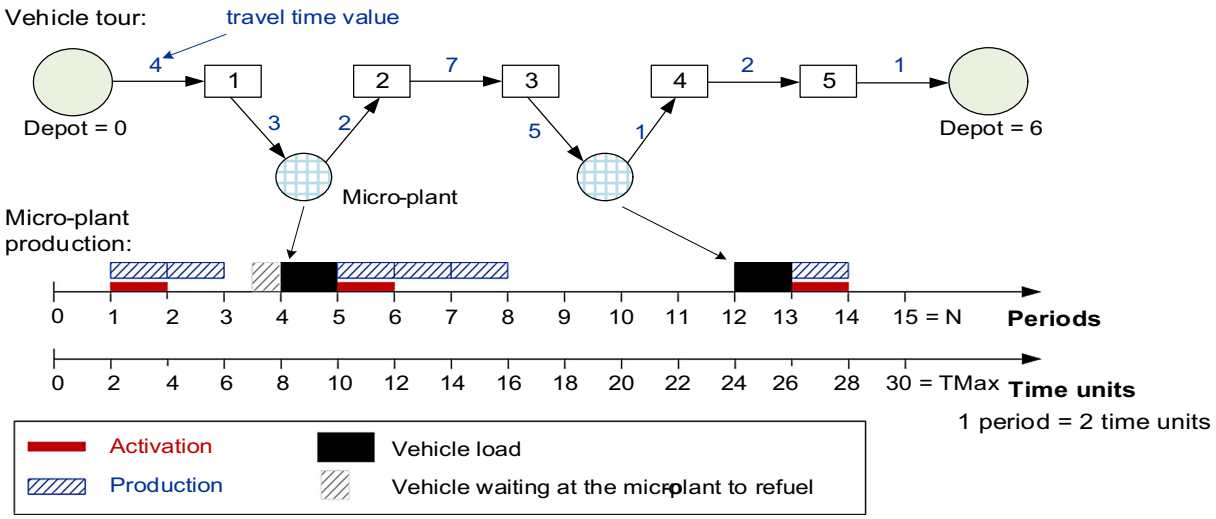

FIGURE 5. A feasible solution related to Figures 2 and 3.

The global resulting activation cost is 3*7 = 21. The time T is equal to 30. The time-dependent production cost is equal to 9. Thus, the global cost is equal to $21 + 9 + 30 = 60$.

Remark 2.1. We focus here on synchronization mechanisms and so consider a deterministic version of our problem. Still, in practice, a key issue is about the uncertainty related to the production ratio $R_i, i = 0, \ldots, N-1$. This issue will be addressed in a future work.

The following Table 1 summarizes the input data for the SMEPC problem.

2.2. A Mathematical Programming oriented formulation

Though *Integer Linear Programming* is not well-fitted to SMEPC handling, we first propose a Mathematical Programming oriented formulation, which allows to clearly identify main variables and constraints, and which comes as follows:

- **Production variables.**
  - $z = (z_i, i = -1, \ldots, N-1)$, with $\{0, 1\}$ values: $z_i = 1 \sim$ the micro-plant is active during period $i (i = -1$ corresponds to a fictitious period).
  - $y = (y_i, i = 0, \ldots, N-1)$, with $\{0, 1\}$ values: $y_i = 1 \sim$ the micro-plant is activated at the beginning of period i.
  - $V^{\text{Tank}} = (V_i^{\text{Tank}}, i = 0, \ldots, N-1)$, with non negative integer values: V_i^{Tank} is the H_2 load of the micro-plant tank at the beginning of period i.
  - $\delta = (\delta_i, i = 0, \ldots, N-1)$, with $\{0, 1\}$ values: $\delta_i = 1 \sim$ the vehicle is refueling during i.
  - $L = (L_i, i = 0, \ldots, N-1)$, with non negative integer values: in case $\delta_i = 1, L_i$ is the quantity of H_2 loaded by the vehicle during period i.
- **Vehicle variables.**
  - $x = (x_j, j = 0, \ldots, M)$, with $\{0, 1\}$ values: $x_j = 1 \sim$ the vehicle refuels while traveling from station j to station $j + 1$.
  - $L^* = (L_j^*, j = 0, \ldots, M)$, with non negative integer values: if $x_j = 1, L_j^*$ is the H_2 quantity loaded by the vehicle while traveling from j to $j + 1$.
  - $T = (T_j, j = 0, \ldots, M + 1)$, with non negative integer values: T_j is the time when the vehicle at j.

TABLE 1. Input data for the SMEPC 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: required time to go from station j to station $j+1$
d_j: required time to go from station j to the micro-plant
d_j^*: required time to go from the micro-plant to station j
e_j: required energy to go from station j to station $j+1$
ε_j: required energy to go from station j to the micro-plant
ε_j^*: 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
$P_i = [p.i, p.(i+1)[$: time interval related to production period i
R_i: production rate related to period i
$Cost_i^V$: production cost related to period i

- $T^* = (T_j^*, j = 0, \ldots, M+1)$, with non negative integer values: if $x_j = 1, T_j^*$ is the time when the vehicle starts refueling while traveling from j to $j+1$.
- $V^{\text{Veh}} = (V_j^{\text{Veh}}, j = 0, \ldots, M+1)$, with non negative integer values: V_j^{Veh} is the H_2 load of the vehicle tank when the vehicle arrives in j.

Those variables will be constrained as follows (we use here ILP: *Integer Linear Program* formalism, and explain the way some of those constraints must be understood as the linearization of logical implications *Big M* technique):

– **Objective function.** Minimize

$$\sum_{i=0,\ldots,N-1} \left(Cost^F.y_i + Cost_i^V.z_i\right) + \alpha.T_{M+1}.$$

– **Production constraints.**
- For any $i = 0, \ldots, N-1$:
  - $-y_i + z_i \geq 0$;
  - $y_i + z_{i-1} \leq 1$;
  - $z_i - z_{i-1} \leq y_i$.

  Explanation. Those 2 constraints express the logical equivalence: $y_i = 1 \Leftrightarrow (z_i = 1 \wedge z_{i-1} = 0)$.
- For any $i = 0, \ldots, N-1 : z_i + \delta_i \leq 1$.
- $z_0 = y_0$.
- $V_0^{\text{Tank}} = H_0; V_N^{\text{Tank}} \geq H_0$.
- For any $i = 0, \ldots, N-1 : V_i^{\text{Tank}} \leq C^{\text{MP}}$.
- For any $i = 0, \ldots, N-1$:
  - $V_{i+1}^{\text{Tank}} = V_i^{\text{Tank}} + z_i.R_i - L_i$;
  - $L_i \leq \delta_i.C^{\text{MP}}$.

Explanation. Those 3 constraints linearize the quadratic constraint: $V_{i+1}^{\text{Tank}} = V_i^{\text{Tank}} + z_i.R_i - \delta_i.L_i$.

– **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)
- For any $j = 0, \ldots, M$:
  - $T_{j+1} - (T_j + t_j) \geq -2.x_j.T\text{Max}$;
  - $T_{j+1} - (T_j^* + p + d_{j+1}^*) \geq -2.T\text{Max}.(1 - x_j)$.

  Explanation. Those 2 constraints linearize, through *Big M* technique, the quadratic constraint:

$$T_{j+1} \geq (1 - x_j).(T_j + t_j) + x_j.\left(T_j^* + p + d_{j+1}^*\right).$$

- For any $j = 0, \ldots, M : T_j^* \geq T_j + d_j$.
- For any $j = 0, \ldots, M$:
  - $L_j^* \leq x_j.C^{\text{Veh}}$;
  - $L_j^* \leq C^{\text{Veh}} + \varepsilon_j - V_j^{\text{Veh}}$;
  - $V_{j+1}^{\text{Veh}} = V_j^{\text{Veh}} - e_j + x_j.(e_j - \varepsilon_j - \varepsilon_{j+1}^*) + L_j^*$.

  Explanation. Those 3 constraints are the translation of the following logical implication:
  - $x_j = 0 \rightarrow V_{j+1}^{\text{Veh}} = V_j^{\text{Veh}} - e_j.$;
  - $x_j = 1 \rightarrow V_{j+1}^{\text{Veh}} = V_j^{\text{Veh}} - \varepsilon_j - \varepsilon_{j+1}^* + L_j^*$.
- $T_{M+1} \leq T\text{Max}$.

Remark 2.2. Constraint (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.

In order to get a complete formulation, we must explain the way the activities of both the vehicle and the micro-plant are synchronized. This requires introducing a *synchronization* variable $U = (U_{i,j}, i = 0, \ldots, N-1, j = 0, \ldots, M)$ with $\{0, 1\}$ values, which will tell us, in case the vehicle decides to refuel between station j and station $j+1$, during which period i it will do it.

– **Synchronization variables:** $U = (U_{i,j}, i = 0, \ldots, N-1, j = 0, \ldots, M)$ with $\{0, 1\}$ values: $U_{i,j} = 1 \sim$ the vehicle is going to refuel during period i while traveling from j to $j+1$.

Then, we complete our SMEPC formulation with the following *synchronization* constraints:

– **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 $j = 0, \ldots, M, T_j^* \geq \Sigma_{i=0,\ldots,N-1} p.i.U_{i,j}$.
- For any $i = 0, \ldots, N-1 : L_i \leq \text{Min}(V_i^{\text{Tank}}, \Sigma_{j=0,\ldots,M} U_{i,j}.(C^{\text{Veh}} + \varepsilon_j - V_j^{\text{Veh}}))$. (E2)
- For any $j = 0, \ldots, M : L_j^* = \Sigma_{i=0,\ldots,N-1} U_{i,j}.L_i$.
- For any $j = 0, \ldots, M : \Sigma_{i=0,\ldots,N-1} p.i.U_{i,j} \geq T_j + d_j - 2.T\text{Max}.(1 - x_j)$.
- **Explanation:** This constraint is the linearization, through *Big M* technique, of the following implication: $x_j = 1 \rightarrow \Sigma_{i=0,\ldots,N-1} p.i.U_{i,j} \geq T_j + d_j$.

Remark 2.3. In practice, we should be able to replace the "$\leq$" in constraint (E2) by a "=" symbol, since in most case, a good strategy for the vehicle will make it to refuel as much as possible. Still we must be sure that we avoid producing more than necessary.

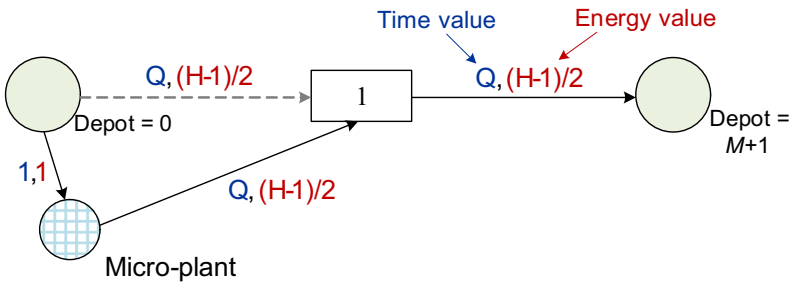

FIGURE 6. Vehicle trip (the vehicle refuels at the micro-plant during $p = 1$ time unit).

3. A Dynamic Programming (DPS) Scheme

We first easily check that, even when $Cost^F$ is null, SMEPC can be reduced to a Knapsack problem, and so we state:

Theorem 3.1. *SMEPC is NP-Hard.*

Proof. Let us consider the following specific instance of SMEPC:

- N is given as an even number $N = 3.Q + 1$, where Q is some parameter supposed to be ≥ 2.
- $p = 1$.
- Scaling parameter α is null.
- *Activation cost* $Cost^F = 0$; *Time-Dependent costs* $Cost_i^V$ *are null as soon as* $i \geq Q$.
- $T\text{Max} = N = 3.Q + 1$.
- Production levels R_i are null as soon as $i \geq Q$.
- Both capacities C^{Veh} and C^{MP} are infinite.
- Initial micro-plant load $H_0 = 0$ and initial vehicle load $E_0 = 1$.
- $M = 1$ and the route $\Gamma = \{Depot, 1, Depot\}$ and:
  - $d_1 = d_1^* = Q; d_0 = d_0^* = 1$;
  - $t_0 = t_1 = Q$;
  - $\varepsilon_1 = \varepsilon_1^* = (H - 1)/2$, where H is some parameter supposed to be ≥ 2;
  - $\varepsilon_0 = \varepsilon_0^* = 1$;
  - $e_0 = e_1 = (H - 1)/2$.

Then, we see that the micro-plant must produce H hydrogen energy units during periods $0, \ldots, Q - 1$, in such a way that vehicle may leave at date $Q - 1$ and achieve its tour in $2Q + 1$ periods (including $p = 1$ time unit to refuel, see Fig. 6) and return to *Depot* with a tank loaded with one hydrogen energy unit. It comes then that under those hypothesizes, minimizing the economic cost of the process means solving the following Knapsack instance:

{Compute a $\{0, 1\}$-valued vector $Z = (Z_0, \ldots, Z_{Q-1})$ such that:
- $\Sigma_{i=0,\ldots,Q-1} R_i.Z_i \geq H$;
- $\text{Min} = \Sigma_{i=0,\ldots,Q-1} Cost_i^V.Z_i$}.

We conclude. □

Still, as we shall see now, SMEPC lies at the frontier of Time-Polynomiality.

3.1. The DPS (dynamic programming scheme)

DPS *time space* and *states*. The *time space* of our DPS is the set Δ of *time pairs* $(i,j), i = 0,\dots,N, j = 0,\dots,M+1$, provided with its standard partial ordering. We may extend this partial ordering into a linear ordering in several ways, providing every time pair $(i,\, j)$ with a successor $\mathrm{Succ}_\Delta(i,j)$, for example by setting:

$$\mathrm{Succ}_\Delta(i,j) = (i+1,j) \text{ if } i \leq N-1 \text{ and } \mathrm{Succ}_\Delta(i,j) = (0,j+1) \text{ else.}$$

We link periods i and stations j by introducing relations ($\ll$, $\gg$, ==) which express the relative positioning of period i and the time T at which the vehicle is at station j. Namely, for any $i \in \{0,\dots,N-1\}$ and $T \in \{0,\dots,T\mathrm{Max}\}$, we set:

- $T \ll i$ if $T < p.i$;
- $T \gg i$ if $T \geq p.(i+1)$;
- $T == i$ if $p.i \leq T < p.(i+1)$.

Those relations are going to help us in defining the state variables of our DPS scheme. As a matter of fact, we say that, for any such a *time pair* (i,j), a *state* is a 4-uple $s = (Z,T,V^{\mathrm{Tank}},V^{\mathrm{Veh}})$, with:

- $Z = 1 \sim$ 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 value in $0,\dots,T\mathrm{Max}$ with the meaning:
  - 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 in j, and must choose between keeping on to $j+1$ or moving to the micro-plant in order to refuel.

Remark 3.2. We see that according to this explanation, relative positioning of T and i with respect to the ==, $\ll$ and $\gg$ relations acts as an implicit state variable, which will help us in identifying the decisions which may be associated with a current time pair $(i,\, j)$ and a current state s, together with their impact.

Decisions. Then a decision D related to a *time pair* $(i,\, j)$ and a *state* $s = (Z,\, T,\, V^{\mathrm{Tank}},\, V^{\mathrm{Veh}})$ 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 as soon as $T == i$: in such a case, $x = 0$ means that the vehicle moves from station j to station $j+1$ without refueling; and $x = 1$ means that it refuels at the micro-plant while traveling from j to $j+1$. In the case when $x = 0$, the induced transition will turn j into $j' = j+1$ and i into $i' = i+1$, except in the case when $T + t_j == i$: in this last case, no real decision is taken with respect to the production and z is equal to 0 by default. If Not $(T == i)$ then x becomes irrelevant and its value is by default equal to 0, no transition being associated with it.
- $\delta = 1 \sim$ 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. This means that $T \ll i$ and that the difference $p.i - T$ is at least equal to the time required in order to move from j to the micro-plant.

Decision is taken at the end of period $i-1$. Before providing the detail of preconditions, transitions and costs related to those states and decisions, we may come back to the example of Section 2.1 and describe the way states and decisions will evolve according to the feasible solution described in Figure 7 below.

Then Table 2 provides us with the evolution of time pairs $(i,\, j)$, states $s = (Z,T,V^{\mathrm{Tank}},V^{\mathrm{Veh}})$, decisions $D = (z,x,\delta)$ and costs $Cost + \alpha.T$, induced by this solution.

Preconditions, transitions and costs related to decisions and states. We are going now to consider time pair $(i,\, j)$ a current state $s = (Z,T,V^{\mathrm{Tank}},V^{\mathrm{Veh}})$, and scan the set of all possible decisions $D = (z,x,\delta)$.

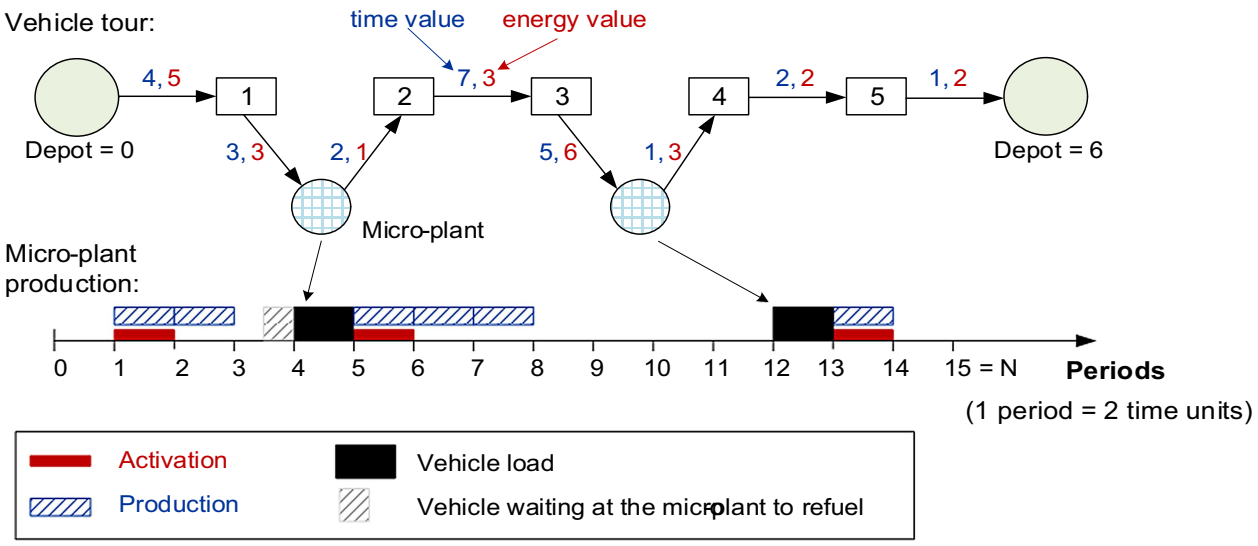

FIGURE 7. A feasible solution related $p = 2, E_0 = 8, H_0 = 4$, $T\text{Max} = 30, Cost^F = 7, C^{\text{MP}} = 15, C^{\text{Veh}} = 15, \alpha = 1$, together with t, d, e, ε, R coefficients as in Figures 2 and 3.

TABLE 2. Simulation of the evolution of $(i, j), s, D$ and $Cost + \alpha.T$ related to Figure 7.

Time pair (i,j)		State $s = (Z, T, V^{\text{Tank}}, V^{\text{Veh}})$				Solution cost W	Decision $D = (z, x, \delta)$		
i	j	Z	T	V^{Tank}	V^{Veh}	$W = Cost + T$	z	x	δ
0	0	0	0	4	8	0 + 0	0	0	0
1	1	0	4	4	3	0 + 4	1	0	0
2	1	1	4	9	3	8 + 4	1	1	0
3	1	1	4	13	3	9 + 4	0	0	0
4	1	0	4	13	3	9 + 4	0	0	1
5	2	0	12	0	12	9 + 12	1	0	0
6	3	1	19	3	9	18 + 19	1	0	0
7	3	1	19	8	9	20 + 19	1	0	0
8	3	1	19	12	9	22 + 19	0	0	0
9	3	0	19	12	9	22 + 19	0	1	0
10	3	0	19	12	9	22 + 19	0	0	0
11	3	0	19	12	9	22 + 19	0	0	0
12	3	0	19	12	9	22 + 19	0	0	1
13	4	0	27	0	12	22 + 27	1	0	0
14	5	1	29	4	10	30 + 29	0	0	0
15	6	0	30	4	8	30 + 30	*	*	*

For every such a decision D we are going to detail the conditions which will ensure the feasibility of D, the resulting transition $((i, j), s) \to ((i', j'), s')$ and the cost CT of this transition.

– $z = 1, x = 0, \delta = 0$: the micro-plant produces during period i while the vehicle keeps on as previously. It requires:
  - $V^{\text{Tank}} + R_i \leq C^{\text{MP}}$;
  - if $T == i$, then $V^{\text{Veh}} - e_j - \varepsilon_{j+1} \geq 0$ and $T + t_j \gg i$;
  - if $T \ll i$ then $p.(i + 1) + p + d^*_{j+1} + \Sigma_{k \geq j+1} t_k \leq T\text{Max}$.

If Not $(T == i)$, then $x = 0$ does not refer to any true decision. We shift from (i, j) to $(i+1, j)$ and resulting *state* is $s' = (1, T, V^{\text{Tank}} + R_i, V^{\text{Veh}})$. Transition cost is $(Cost^F.(1 - Z) + Cost^V_{i.})$.
If $(T == i)$, then the vehicle starts travelling from j to $j + 1$. We shift from (i, j) to $(i + 1, j + 1)$ and resulting *state* is $s' = (1, T + t_j, V^{\text{Tank}} + R_i, V^{\text{Veh}} - e_j)$. Transition cost is $(Cost^F.(1 - Z) + Cost^V_{i.}) + \alpha.t_j$.

– $\underline{z = 1, x = 1, \delta = 0}$: the micro-plant produces during period i and the vehicle decides to move from j to the micro-plant in order to refuel. It requires:
  - $T == i$;
  - $V^{\text{Tank}} + R_i \leq C^{\text{MP}}$;
  - $\text{Sup}(p.(i + 1), T + d_j) + d^*_{j+1} + p + \Sigma_{k \geq j+1} t_k \leq T\text{Max}$.

  We shift from (i, j) to $(i + 1, j)$ and resulting *state* is $s' = (1, T, V^{\text{Tank}} + R_i, V^{\text{Veh}})$. Transition cost is $(Cost^F.(1 - Z) + Cost^V_i)$.
– $\underline{z = 0, x = 0, \delta = 0}$: the micro-plant does not produce during period i and the vehicle keeps on its way. It requires:
  - if $T == i$, then $x = 0$ means that the vehicle is fueled enough in order to move from j to $j + 1$: $V^{\text{Veh}} - e_j - \varepsilon_{j+1} \geq 0$;
  - if $T + d_i \leq p.i$ then the vehicle previously decided to refuel between j and $j + 1$, and is currently located at the micro-plant. Decision $\delta = 0$ is feasible if the vehicle has enough time to keep waiting, that means: $p.(i + 1) + p + d^*_{j+1} + \Sigma_{k \geq j+1} t_k \leq T\text{Max}$.

  If $T \gg i$ or $T \ll i$, 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.
  If $T == i$ then we shift from (i, j) to $(i + 1, j + 1)$ but in the case when $T + t_j == i$: in this case i remains unchanged. In any case resulting *state* is $s' = (0, T + t_j, V^{\text{Tank}}, V^{\text{Veh}} - e_j)$ and transition cost is $\alpha.t_j$.
– $\underline{z = 0, x = 1, \delta = 0}$: the micro-plant does not produce during period i and the vehicle moves towards the micro-plant in order to refuel. It requires:
  - $T == i$;
  - $\text{Sup}(p.(i + 1), T + d_j) + d^*_{j+1} + p + \Sigma_{k \geq j+1} t_k \leq T\text{Max}$.

  Then we shift from (i, j) to $(i + 1, j)$ and resulting *state* is $s' = (0, T, V^{\text{Tank}}, V^{\text{Veh}})$. Transition cost is null.
– $\underline{z = 0, x = 0, \delta = 1}$: the micro-plant does not produce during period i and the vehicle decides to refuel during period i. It requires:
  - $T + d_j \leq p.i$;
  - $\varepsilon_{j+1} + \varepsilon^*_{j+1} \leq \text{Inf}(C^{\text{Veh}}, V^{\text{Tank}} + V^{\text{Veh}} - \varepsilon_j)$.

  Then we shift from (i, j) to $(i + 1, j + 1)$ and resulting *state* is $s' = (0, p.(i + 1) + d^*_{j+1}, V^{\text{Tank}} - \text{Inf}(V^{\text{Tank}}, C^{\text{Veh}} - V^{\text{Veh}} + \varepsilon_j), \text{Inf}(C^{\text{Veh}} - \varepsilon^*_{j+1}, V^{\text{Tank}} + V^{\text{Veh}} - \varepsilon_j - \varepsilon^*_{j+1}))$. Transition cost is $\alpha.(p.(i + 1) + d^*_{j+1} - T)$.

Remark 3.3. In the specific case when the energy amount V^* the vehicle needs to fuel in order to achieve its trip until depot with a load at least equal to initial load E_0 is less than $\text{Inf}(V^{\text{Tank}}, C^{\text{Veh}} - V^{\text{Veh}} + \varepsilon_j)$ then the resulting s' is $(0, T, V^{\text{Tank}} - V^*, V^{\text{Veh}} - \varepsilon_j + V^* - \varepsilon^*_{j+1})$. In such a case, we see that this loading transaction will be the last one.

– $\underline{z = 0, x = 1, \delta = 1}$ as well as $\underline{z = 1, x = 0/1, \delta = 1}$ are forbidden.

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 the value W equal to the smallest possible cost of a sequence of feasible transitions from initial state $s_0 = (0, 0, H_0, E_0)$ at time $(0, 0)$ to state $s = (Z, T, V^{\text{Tank}}, V^{\text{Veh}})$ at time (i, j). Then we implement our DPS according to a forward driven strategy. For any current *time pair* (i, j) we denote by $S(i, j)$ what the *state subset* related to (i, j) and which is a set of 3-uples $(s, W, Father)$ where W is the above defined value and Father means the pair

$((i_f, j_f), s_f)$ where s_f is the state related to the time pair (i_f, j_f) which allowed us to reach the state s at time pair (i, j).

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 the related feasible decision set $Dec((i, j), s)$. Then, for every such a decision $D = (z, x, \delta)$, 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 cost of the transition from $((i, j), s) \rightarrow ((i', j'), s')$ and W is the value associated with (i, j) and s:

- If *state* s' does not appear yet in $S(i', j')$ then we insert it into $S(i', j')$, together with value $W + \text{CT}$.
- If *state* s' already appears in $S(i', j')$ with a value $W^* > W + \text{CT}$, then $W + \text{CT}$ becomes the value associated with (i', j') and s'.
- If *state* s' already appears in $S(i', j')$ with a value $W^* \leq W + \text{CT}$, then we discard s'.

We denote by *DPS-SMEPC* the dynamic network algorithm designed this way, whose full description may be summarized as follows.

DPS-SMEPC algorithm.

Input: $N, M,$ $T\text{Max}, H_0, E_0, C^{\text{MP}}, C^{\text{Veh}}$
Output: *Current-Sol, Current-Value*

Initialization: $i \leftarrow 0; j \leftarrow 0; W \leftarrow 0; S(0, 0) \leftarrow \{(s_0 = (0, 0, H_0, E_0), W = 0,$ *Father* $=$ Undefined$)\}$;
Current-Sol $\leftarrow$ Undefined; *Current-Value* $\leftarrow +\infty$;

Main Loop:
While $j + i \leq N + M + 1$ **do**
For $(s,$ $W,$ *Father*$)$ in the list $S(i, j)$ **do**
Generate the feasible *Decision Set* $Dec((i, j), s)$;
For $D = (z, x, \delta)$ in $Dec((i, j), s)$ **do**
Compute resulting *time pair* (i', j') and *state* $s' = (Z', T', V^{\text{Tank}'}, V^{\text{Veh}'})$,
together with transition cost CT;
If some 3-uple $(s', W', Father')$ already appears in $S(i', j')$ and if $W' > W + \text{CT}$ **then**
Replace $(s', W', Father')$ by $(s', W + \text{CT}, (s, i, j))$;
Else
Insert $(s', W + \text{CT}, (s, i, j))$ into $S(i', j')$; (I1)
EndIf
If $(j' = M + 1)$ & $(T' \leq T\text{Max})$ & $(V^{\text{Tank}} \geq H_0)$ & $(V^{\text{Veh}} \geq E_0)$ & $(Current\text{-}Value > W + \text{CT})$
then
Current-Value $\leftarrow W + \text{CT}$; *Current-Sol* $\leftarrow (s', (s, i, j))$;
EndIf
EndFor
EndFor
$(i,\ j) \leftarrow \text{Succ}_\Delta\ (i, j)$; (I2)
EndWhile

The solution is the sequence of decisions induced by the father field of *Current-Sol.*

Explanation. Instructions (I1) and (I2).

- As shall be discussed in next Section 4, instruction (I1) has to be strengthened by filtering devices, which are going to allow us to *kill* $(s',$ $W',$ *Father*$')$ in case some kind of infeasibility may be forecasted, or in case we may check that keeping on with the trajectory defined by (i', j') and $(s',$ $W',$ *Father*$')$ will not yield a better solution than an already existing one.
- Instruction (I2) tells us about the way we scan the time space Δ, in a way which is consistent with its standard partial ordering. Several functions Succ_Δ may be involved, which scan Δ according to rows i, to columns j, or to diagonal layers.

Complexity of *DPS-SMEPC*. The size of the time space of *DPS-SMEPC* is in $O(N.M)$. The size of its state space is in $O(2.T\text{Max}.C^{\text{Veh}}.C^{\text{MP}})$. The size of the decision space related to given time pair (i, j) and given state s is no more than 8. Testing the feasibility of such a decision and computing resulting state can be performed in constant time. It comes that the (worst case) time complexity of *DPS-SMEPC* is in $O(N.M.T\text{Max}.C^{\text{Veh}}.C^{\text{MP}})$. This statement raises the issue of the control of values $T\text{Max}$, C^{Veh}, C^{MP}, and provides the motivation for Sections 3.2 and 4, respectively related to the design of some theoretical PTAS: *Polynomial Time Approximation Scheme* and the implementation of some practical filtering devices.

3.2. A Polynomial Time Approximation Scheme

DPS-SMEPC may be in trouble as soon as M and N are large: every state is a 4-uple, with T, V^{Veh} and V^{Tank} with potentially large values. Still, we may notice that if we consider $T\text{Max}$, C^{MP} and C^{Veh} as bounded by polynomial functions of N and M, then *DPS-SMEPC* becomes time-polynomial since the number of states it involves also becomes bounded by a polynomial function of N and M.

As a matter of fact, we may go further and introduce a FPTAS like rounding scheme as follows.

Rounding States inside the *DPS-SMEPC* Scheme. L being some integer, then, for any integer n with binary decomposition $n = a_0 + a_1.2 + \ldots + a_q.2^q$, we 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 $Round^*(n, L) = a_{q-K}.2^{q-L} + \ldots + a_q.2^q + 2^{q-L}$.

We say that two integral numbers n and m are equivalent *modulo the L largest bits* if $Round(n, L) = Round(m, L)$.

Let us come back now to our *DPS-SMEPC* algorithmic scheme. We first assume that all inputs d, t, e, ε, R_i, $Cost^F$, $Cost_i^V$, $T\text{Max}$, C^{MP} and C^{Veh} of our SMEPC model are integers. L being an integral number, $s_1 = (Z, T_1, V_1^{\text{Tank}}, V_1^{\text{Veh}})$ and $s_2 = (Z, T_2, V_2^{\text{Tank}}, V_2^{\text{Veh}})$ being two states of our *DPS-SMEPC* algorithmic scheme, we say by extension that they are equivalent *modulo the L largest bits* if $Round(T_1, L) = Round(T_2, L)$, $Round(V_1^{\text{Tank}}, L) = Round(V_2^{\text{Tank}}, L)$, $Round(V_1^{\text{Veh}}, L) = Round(V_2^{\text{Veh}}, L)$.

The DPS-SMEPC (K) Approximation Scheme. Then, being some integer $K \geq 1$, we turn the *DPS-SMEPC* algorithm of previous Section 3.1 into a parametrized algorithm *DPS-SMPEPC*(K) by proceeding as follows:

- We compute $L(N, M)$, with minimal value such that $2^{L(N,M)} \geq N + M + 1$.
- We extend the notion of state, by considering that any state s is a 5-uple $(Z, \Omega, T, V^{\text{Tank}}, V^{\text{Veh}})$ associated to a time pair (i, j), where Ω tells us whether $T \ll i(\Omega = 0), T == i(\Omega = 1), (T \gg i) \wedge (T + d_j > p.i)(\Omega = 2), (T \gg i) \wedge (T + d_j \leq p.i)(\Omega = 3)$: This clearly means that we want to take into account here the fact (see Rem. 3.2) that relative positioning of T and i through relations $\ll, \gg$ and $==$ acts as an implicit state variable. Because of the rounding effects, which are likely to perturb those relations, we introduce variable Ω, whose purpose is to explicit the information provided by those relative positioning.
- We do in such a way that, at any iteration of the main loop of our algorithm, the set $S(i, j)$ does not contain two 3-uples $(s_1, W_1, Father_1)$ and $(s_2, W_2, Father_2)$ such that respectively s_1 and s_2, as well as W_1 and W_2, are equivalent *modulo the* $K + 2.L(N, M)$ *largest bits*; We do it while giving priority to the 3-uple $(s_q, W_q, Father_q), q = 1, 2,$ related to the smallest W_q value.
- We replace initial values H_0 and E_0 by respectively $Round^*(H_0, K + 1)$ and $Round^*(E_0, K + 1)$.
- We relax the time capacity constraint by replacing $T\text{Max}$ by $Round^*(T\text{Max}, K)$; By the same way we relax the H_2 capacity constraints related to both the micro-plant and the vehicle by replacing capacities C^{MP} and C^{Veh} respectively by $Round^*(C^{\text{MP}}, K)$ and $Round^*(C^{\text{Veh}}, K)$: this means that (i, j) being current *time pair*, and s being current *state*, we compute *Decision Set* $Dec((i, j), s)$ (Instruction (I3) below) in such a way that the feasibility of any decision $D = (z, x, \delta)$ is checked with respect to the time capacity $Round^*(T\text{Max}, K)$ and that H_2 capacity constraints are checked with respect to H_2 capacities $Round^*(C^{\text{MP}}, K)$ and $Round^*(C^{\text{Veh}}, K)$.

DPS-SMPEPC(K) may be summarized as follows:

DPS-SMEPC (K) algorithm.

Input: $N, M,\ T\text{Max},\ H_0, E_0, C^{\text{MP}}, C^{\text{Veh}}, K$
Output: *Current-Sol, Current-Value*

Initialization: $L \leftarrow L(N,M); i \leftarrow 0; j \leftarrow 0; W \leftarrow 0; S(0,0) \leftarrow \{(s_0 = (0,0,H_0,E_0), W = 0, \textit{Father} = \text{Undefined})\}$;
$T\text{Max} \leftarrow \textit{Round}^*(T\text{Max}, K)$; $C^{\text{MP}} \leftarrow \textit{Round}^*(C^{\text{MP}}, K)$; $C^{\text{Veh}} \leftarrow \textit{Round}^*(C^{\text{Veh}}, K)$;
$H_0 \leftarrow \textit{Round}^*(H_0, K+1)$; $E_0 \leftarrow \textit{Round}^*(E_0, K+1)$;
Current-Sol $\leftarrow$ Undefined; *Current-Value* $\leftarrow +\infty$;

Main Loop:
While $j + i \leq N + M + 1$ **do**
For $(s, W, \textit{Father})$ in the list $S(i,j)$ **do**
Generate the feasible *Decision Set* $\textit{Dec}((i,j), s)$; (I3)
For $D = (z, x, \delta)$ in $\textit{Dec}((i,j), s)$ **do**
Compute resulting *time pair* (i', j') and *state* $s' = (Z', T', V^{\text{Tank}'}, V^{\text{Veh}'})$, together with transition cost CT;
If some 3-uple $(s'', W'', \textit{Father}'')$ already appears in $S(i', j')$ such that:
– Respectively s' and s'', as well as $W + \text{CT}$ and W'', are equivalent *modulo the* $K + L$ *largest bits*
– $W + \text{CT} < W''$,
then
Replace $(s'', W'', \textit{Father}'')$ by $(s', W + \text{CT}, (s, i, j))$;
Else
Insert $(s', W + \text{CT}, (s, W, \textit{Father}))$ into $S(i', j')$; (I1)
EndIf
If $(j' = M+1)\&(T' \leq T\text{Max})$ & $(V^{\text{Tank}} \geq H_0)$ & $(V^{\text{Veh}} \geq E_0)$ & $(\textit{Current-Value} > W + \text{CT})$ **then**
Current-Value $\leftarrow W + \text{CT}$; *Current-Sol* $\leftarrow (s', (s, i, j))$;
EndIf
EndFor
EndFor
$(i,j) \leftarrow \text{Succ}_\Delta(i,j)$; (I2)
EndWhile

Then we may state.

Theorem 3.4 (Polynomial Time Approximation Scheme)**.** *K being fixed, DPS-SMEPC(K) is time-polynomial. Besides, for any value* $\varepsilon > 0$*, we may choose K large enough in such a way that in case SMEPC admits an optimal solution with value* W^{Opt}*, then DPS-SMEPC(K) yields a solution which is feasible with regards to initial values* $(1+\varepsilon/2).H_0$ *and* $(1+\varepsilon/2).E_0$*, threshold values* $(1+\varepsilon).C^{\text{MP}}$, $(1+\varepsilon).C^{\text{Veh}}$ *and* $(1+\varepsilon).T\text{Max}$ *and whose cost value Current-Value is no larger than* W^{Opt}*.*

Proof. K being fixed, the fact that algorithm *DPS-SMEPC*(K) is time-polynomial comes the same way as when $T\text{Max}$, C^{MP} and C^{Veh} are supposed to be bounded by polynomial functions of N and M: the number of possible 3-uples $(s, W, \textit{Father})$ in the list $S(i,j)$ which we deal with at every iteration of the main loop is bounded by a polynomial function of N, M and the encoding size of $T\text{Max}$, C^{MP} and C^{Veh}. By the same way, ε being given, we see that if K is large enough, then the relative error $(\textit{Round}^*(T\text{Max}, K) - T\text{Max})/T\text{Max}$ induced by replacing $T\text{Max}$ by $\textit{Round}^*(T\text{Max}, K)$ does not exceed ε. The same thing holds for capacities C^{Veh} and C^{MP}.

In order to achieve the proof of Theorem 3.4, we need now to consider an optimal solution Sol^{Opt} given together with its value W^{Opt}. The solution Sol^{Opt} may be associated with a sequence of time values $(i_h j_h) h =$

$0, \dots H \leq N+M$ a sequence of states s_0, $s_1 \dots s_H$ with related value $W_0, \dots, W_H$ and a sequence of decisions $D_0, \dots D_{H-1}$ which induces transitions $((i_h, j_h), s_h), \rightarrow ((i_{h+1}, j_{h+1}), s_{h+1}), h = 0 \dots H-1$ and to check that if K is large enough *DPS-SMEPC*(K) computes a solution *Current-Sol* which is feasible with regards to threshold values $(1+\varepsilon).C^{\mathrm{MP}}, (1+\varepsilon).C^{\mathrm{Veh}}$ and $(1+\varepsilon).T\mathrm{Max}$ and whose cost value *Current-Value* is no larger than $W^{\mathrm{Opt}}+\varepsilon$. In order to do it, we fix K and prove by induction on h that, for any $h = 0, \dots, H$:

– There exists in the set $S(i_h, j_h)$ computed by *DPS-SMEPC*(K) some 3-uple $(s = (Z, T, V^{\mathrm{Tank}}, V^{\mathrm{Veh}}), W,$ *Father*$)$ such that, if we set $s_h = (Z_h, T_h, V_h^{\mathrm{Tank}}, V_h^{\mathrm{Veh}})$: (E3)
- $W \leq W_h$;
- $Z_h = Z; T \leq T_h.(1 + h.2^{-K}/(N+M))$; (E3-1)
- $V^{\mathrm{Tank}}.(1 + h.2^{-K}/2(N+M)) \leq V_h^{\mathrm{Tank}} \leq V^{\mathrm{Tank}}.(1 + h.2^{-K}/(N+M))$; (E3-2)
- $V^{\mathrm{Veh}}.(1 + h.2^{-K}/2(N+M)) \leq V_h^{\mathrm{Veh}} \leq V^{\mathrm{Veh}}.(1 + h.2^{-K}/(N+M))$. (E3-3)

We suppose it is true for a given h and try to apply the decision D_h to state s_h. We derive from (E3-1) that applying D_h will not contradict the *Round*$^*(T\mathrm{Max}, K)$ threshold. Because of (E3-2), D_h will not make the load in the micro-plant become negative or exceed *Round*$^*(C^{\mathrm{MP}}, K)$. By the same way, (E3-3) implies that D_h will not make the load in the vehicle be negative or exceed *Round*$^*(C^{\mathrm{Veh}}, K)$. It comes that this decision is going to be feasible Then we see that state $s' = (Z', T', V^{\mathrm{Tank}'}, V^{\mathrm{Veh}'})$ resulting at time value (i_{h+1}, j_{h+1}) from application of D_h to state s and related value W' are going to be such that: (E3')
- $Z_{h+1} = Z'; T' \leq T_{h+1}.(1 + h.2^{-K}/(N+M))$; (E3-1′)
- $V^{\mathrm{Tank}'}.(1 + h.2^{-K}/2(N+M)) \leq V_{h+1}^{\mathrm{Tank}} \leq V^{\mathrm{Tank}'}.(1 + h.2^{-K}/(N+M))$; (E3-2′)
- $V^{\mathrm{Veh}'}.(1 + h.2^{-K}/2(N+M))) \leq V_{h+1}^{\mathrm{Veh}} \leq V^{\mathrm{Veh}'}.(1 + h.2^{-K}/(N+M))$. (E3-3′)
- $W' \leq W_{h+1}$.

But (E3-1′, E3-2′, E3-3′) combined with propagation rules related to relative errors imply that if $(s^*, W^*,$ *Father*$^*)$ is the element equivalent to (s', W') *modulo the* $K+L$ *largest bits* which remains in $S(i_{h+1}, j_{h+1})$ according to *DPS-SMEPC*(K) then we have: (E3*)
- $Z_{h+1} = Z*; T^* \leq T_{h+1}.(1 + (h+1).2^{-K}/(N+M))$; (E3-1*)
- $V^{\mathrm{Tank}} * .(1 + (h+1).2^{-K}/2(N+M)) \leq V_{h+1}^{\mathrm{Tank}} \leq V^{\mathrm{Tank}} * .(1 + (h+1).2^{-K}/(N+M))$; (E3-2*)
- $V^{\mathrm{Veh}} * .(1 + (h+1).2^{-K}/2(N+M)) \leq V_h^{\mathrm{Veh}} \leq V^{\mathrm{Veh}} * .(1 + (h+1).2^{-K}/(N+M))$. (E3-3*)
- $W* \leq W_{h+1}$.

We deduce (E3) for any $h = 0, \dots, H$. We conclude by choosing K in such that $2^K \geq (N+M+1)/\varepsilon$. □

4. Filtering devices

In spite of above result, we face an issue related to the large number of *states* when N and M increase, and so we must think into filtering devices. Clearly, the following *strong dominance* rule may be applied:

– ***Strong Dominance*** **rule:** for a given *time pair* (i, j), if 2 related *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:
- $W_1 \leq W_2; T_1 \leq T_2; Z_1 \geq Z_2; V_1^{\mathrm{Tank}} \geq V_2^{\mathrm{Tank}}; V_1^{\mathrm{Veh}} \geq V_2^{\mathrm{Veh}}$;
- at least one among above inequalities is strict;

then s_1 *strongly dominates* s_2, and we *kill* s_2 (*i.e.* we remove it from the list $S(i, j)$).

This *Strong Dominance* rule has little filtering power, since it is too restrictive. Still, other filtering devices may be implemented, close to the ones which were introduced in [23] by Lozano and Medeglia in their *Pulse* algorithm for the *Constrained Shortest Path* problem, and which are: *Heuristic Weak Dominance* rule, *Logical* filtering rules and *Upper/lower Bound* based filtering rule.

4.1. The weak dominance rule

The idea here is to design a flexible device which will help us in keeping the number of *states* under some threshold NS. Let A, B, C, D, E be some positive parameters. We proceed in such a way that for any time pair (i, j), we only keep the NS *states* with smallest: $A.W + B.T - C.Z_1 - D.V^{\text{Tank}} - E.V^{\text{Veh}}$. Choosing *ad hoc* values (A, B, C, D, E) is an issue by itself. We consider the following *money conversion* values:

$(A, B, C, D, E) = (1, \alpha, Cost^F, Cost^V, Cost^V)$, where $Cost^V$ is the average of $Cost^V_{i.}$, values, $i = 0, \ldots, N-1$.

4.2. Logical filtering rules

Given a time pair (i, j), and a related state $s = (Z, T, V^{\text{Tank}}, V^{\text{Veh}})$, the idea here is to anticipate the non feasibility of a trajectory starting from state s at time (i, j). This non feasibility might be related either to a lack of time (impossible to achieve the vehicle tour before deadline TMax) or to energy production (impossible to get enough fuel in order to achieve the tour). More precisely, for any $j = 0, \ldots, M$, we get a rough estimation of both energy and time required in order to allow the vehicle to return from j to *Depot* by proceeding as follows:

– Apply the following process:
  $H \leftarrow \Sigma_{k \geq j}\ e_j + E_0$; Not *Stop*; *Refuel* $\leftarrow 0$;
  While Not *Stop* **do**;
    Refuel-Aux $\leftarrow \lfloor H/C^{\text{Veh}} \rfloor$;
    If *Refuel-Aux* = *Refuel* then *Stop* **Else**
      $H \leftarrow H + \Sigma_{q=Refuel,\ldots,Refuel-Aux}$; *Refuel* $\leftarrow$ *Refuel-Aux*.

The vehicle will have to load at least $H - V^{\text{Veh}}$ in order to achieve its journey from j and perform at least *Refuel* refueling transactions.

– Denote by $\Omega_0, \ldots, \Omega_{M-j}$, the quantities $(\varepsilon_k + \varepsilon^*_{k+1} - e_k), M \geq k \geq j$, labeled in increasing order.
– Denote by $\Delta_0, \ldots, \Delta_{M-j}$, the quantities $(d_k + d^*_{k+1} - t_k), M \geq k \geq j$, labeled in increasing order: for any station j, we denote $\Delta_j = \Sigma_{k \geq j} t_k + \Sigma_{q=0,\ldots,Refuel-1} \Delta_q$: the vehicle needs at least Δ_j time units in order to achieve its trip from j to *Depot*.
– Denote by *Prod-Max*(i) the quantity $\Sigma_{k \geq i} R_k$: the micro-plant cannot produce more than *Prod-Max*(i) from time *p.i*.

This allows us to state the following filtering rules:

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

If we refer to the description of the *DPS-SMEPC* algorithm of previous Section 3.1, then we see that instruction (I1): Insert $(s', W + \text{CT}$, $(s, W,$ *Father*$))$ into $S(i', j')$, has then to be turned into the following instruction (I1_1):

Instruction I1_1:
Successively apply to s' the *Strong Dominance* rule, the *Makespan Based* rule and the *Energy Based* rule.
If s' has not been *killed* then insert $(s', W + \text{CT}, (s, W, \textit{Father}))$ into $S(i', j')$; (I1_1)

Remark 4.1 (About (worst case) complexity)**.** Though logical filtering devices as those applied in (II_1) will have a strong impact on the running times of the algorithm, it is not clear that they have also an impact on its theoretical worst case complexity. In order to prove such an impact, we should be able to quantify the number of states s which are going to survive in state set $S(i, j)$ for every time pair (i, j), and this is really a difficult issue. It comes that, at the current time, we must consider that the theoretical worst case complexity of *DPS-SMEPC* strengthened by instruction (I1_1) is the same as the complexity which we got in Section 3.1. The same remark will hold in the case of upper/lower bound based filtering rules.

4.3. Upper/lower bound based filtering rule

We easily characterize, for any energy amount V and any period number i, the minimal cost *Cost-Min*(i, V, Z) required from the micro-plant to produce V energy units between time $p.i$ and time $T\text{Max}$, Z denoting the state of the micro-plant at the end of period $i-1$:

Cost-Min$(N, V, Z) = 0$ if $V = 0$ and undefined else;
Cost-Min$(i, V, Z) = \text{Inf}\left[\textit{Cost-Min}(i+1, V, 0), \textit{Cost-Min}\,(i+1, V - R_i, 1) + \left(\textit{Cost}^F.(1-Z) + \textit{Cost}_{i.}^V\right)\right]$.

We implement it through backward driven DPS, and keep in memory the values *Cost-Min*(i, V, Z), provided that the scope of values for V is not too large (else we perform some rounding of V values).

Given now a *time pair* (i, j) and some related state $s = (Z, T, V^{\text{Tank}}, V^{\text{Veh}})$ with value W. Then we get a lower bound LB of the best SMEPC value which may be derived from (i, j) and s by setting: $\text{LB}((i, j), s) = \alpha.\Delta_j +$ *Cost-Min*$(i,\ (H - V^{Tank)+}, Z)$.

If we suppose now that we are provided with some SMEPC feasible value *Current-Value* then we may implement the following filtering rule:

(3) *Upper/Lower Bound Based* filtering rule: if $W + \text{LB}((i, j),\ s) \geq$ *Current-Value*, then *kill state* $s = (Z, T, V^{\text{Tank}}, V^{\text{Veh}})$, related to time pair (i, j).

If we refer now to the the *DPS-SMEPC* algorithm of previous Section 3.1, then we replace instruction: (I1): Insert $(s', W + \text{CT},\ (s, W,\ \textit{Father}))$ into $S(i', j')$, by the following instruction (I1_2):

Instruction I1_2:
Successively apply to s' the *Strong Dominance* rule, the *Makespan Based* rule and the *Energy Based* rule.
If s' has not been *killed* then apply to s' the *Upper/Lower Bound* Rule.
If s' has not been *killed* then insert (s', $W + \text{CT}$, $(s, W,\ \textit{Father})$) into $S(i', j')$. (I1_2)

4.4. Computing an upper bound through a greedy adaptation of DPS-SMEPC

Turning *DPS-SMEPC* into a greedy algorithm can be done by using our dynamic programming scheme and performing a forward driven route in the SMEPC state network according to the best LB value. As a matter of fact, we only need to apply the *weak dominance* filtering mechanism of previous Section 4.1 with $\text{NS} = 1$. Once it has been done, we shall modify the Initialization step of the *DPS-SMEPC* algorithm of previous Section 3.1, by setting: *Current-Value* ← *Val-Greedy*; *Current-Sol* ← *Sol-Greedy*.

As usual, it will be possible, by introducing some random sorting device inside the procedure in charge of choosing an *ad hoc* decision D, to make this greedy algorithm work in a non deterministic way and perform it according to the *multi-start* paradigm.

Still, while adapting *DPS-SMEPC* in order to restrict the *weak dominance* rule of Section 4.1 to the case when $\text{NS} = 1$, we must take care of avoiding deadlock strategies which would make both the micro-plant and the vehicle wait for better prices $Cost_i^V$ and production values R_i. Since our ability to anticipate inconsistencies related to a lack of time or a lack of energy refueling capacity only derives from approximation devices, there is a risk that such waiting strategies may make our process fail in computing some feasible solution. So, in order to make this risk of failure decrease, we forbid:

- decision $(z = 0, x = 1)$ related to situations when $T == i$ and the micro-plant tank is not loaded enough in order to allow to fully refuel the vehicle;
- decision $(z = 0, x = 0)$ related to situations when $T \ll i$ and $(p.i - T) \leq d_j$, and the micro-plant tank is not loaded enough in order to allow to fully refuel the vehicle;
- decision $(z = 0, x = 0, \delta = 0)$ related to situations when $T \ll i$ and $(p.i - T) \geq d_j$.

Then the *Greedy-SMEPC* greedy algorithm comes as follows:

Greedy-SMEPC **algorithm**

Input: $N, M,\ T\text{Max},\ H_0, E_0, C^{\text{MP}}, C^{\text{Veh}}$
Output: *Current-Sol, Current-Value*

Initial *state* $s \leftarrow (0,\ 0,\ H_0, E_0)$; Related value $W \leftarrow 0$; *Time pair* $(i, j) \leftarrow (0,\ 0)$; Not *Stop*;
While Not *Stop* **do**
  Set current *state* $s = (Z, T, V^{\text{Tank}}, V^{\text{Veh}})$, related value $= W$ and current *time pair* $= (i, j)$;
  Pick up a feasible (in the sense of Sect. 3) decision (z, x, δ) such that:
  – Decision (z, x, δ) is not forbidden according to above prohibition rules;
  – Resulting state $s_1 = (Z_1, T_1, V_1^{\text{Tank}}, V_1^{\text{Veh}})$ is not *killed* at resulting time (i_1, j_1) by the above logical filtering devices;
  – $W+$ *Transition-Cost* $+ \text{LB}((i_1, j_1), s_1)$ is the smallest possible, where *Transition-Cost* means the cost of the transition induced by applying (z, x, δ) to s at time pair (i, j)
  If (z, x, δ) does not exists **then**
    Stop (*Failure*); *Sol-Greedy* $\leftarrow$ *Undefined*; *Val-Greedy* $\leftarrow +\infty$;
  Else If $j_1 = M+ 1$ **then**
    Stop (*Success*);
    Derive *Val-Greedy* $= W+$ *Transition-Cost* and related solution *Sol-Greedy* $= (s_1,\ (s, i, j))$;
    Else Apply decision (z, x, δ): Update i, j, s and W accordingly.
    EndIf
  EndIf
EndWhile

Clearly we may make this greedy algorithm work in a non deterministic way and perform it according to the *multi-start* paradigm.

5. Numerical experiments

Purpose. What we want here is to get an evaluation of the impact of the filtering devices described in Section 4 and the quality of the greedy procedure described in Section 4.2. This impact is two-sided: (1) It makes decrease the number of states which are handled throughout the process. (2) It may induce a gap to optimality.

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 GB RAM and Visual Studio 2017 compiler.

Instances. Instances are generated as follows:

- We fix N, M and p and randomly generate stations j and *Depot* and the *Micro-Plant* as point of the R^2 space.
- Then d_j, d_j^* and $t_j, e_j, \varepsilon_j, \varepsilon_j^*$ respectively corresponds to Euclidean distance and Manhattan distance roundings, in such a way they take integral values.
- Then we fix $C^{\text{MP}}, C^{\text{Veh}}$ and $T\text{Max}$, 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 case of this experimental section, we use a package of 15 instances, while making vary N, M, p, as well as the economic costs $Cost^F$ and $Cost_i^V, i = 0, \ldots, N-1$, and fixing $C^{\text{Veh}} = 20$, and $C^{\text{MP}} = 50$, $T\text{Max} = 200$. Main characteristics of the 15 instances package, where $Mean\_Cost^V$ denotes the average value $Cost_i^V$ for $i = 0, \ldots, N-1$, may be summarized inside the following Table 3.

Instance website. Instances are available at the following address:
https://perso.limos.fr/~hetoussa/doc/Instances_RAIRO_2021.zip.

Outputs. For any instance, optimality is achieved through exact *DPS-SMEPC* and so:

TABLE 3. Instances.

Instance	M	N	p	$Cost^F$	$Mean\_Cost^V$
1	4	15	4	1	1.0
2	8	25	4	21	1.1
3	8	26	4	12	4.1
4	8	26	4	15	2.8
5	8	27	4	22	2.1
6	8	30	4	22	3.5
7	10	17	4	9	5.4
8	10	19	4	9	2.1
9	10	20	4	8	2.3
10	10	36	2	25	4.6
11	10	50	4	7	3.3
12	10	57	2	13	1.9
13	10	78	1	3	4.2
14	12	32	4	31	4.7
15	14	45	4	22	6.2

(A) We first run a multi-start version of *Greedy-SMEPC* with 50 replications, and compute related gap G_G to optimality (in %), together with related CPU time G_T (in seconds).

(B) Next we run heuristic *DPS-SMEPC*(NS), with NS as described in Section 4.1 and ranging from 20 to 100, in such a way that no more than NS states are allowed for every time pair (i, j). *DPS-SMEPC*(NS) involves the filtering rules described in Sections 4.2 and 4.3. For every instance, we compute gap W_G(NS) to optimality (in %), together with related CPU time W_G(NS) (in seconds).

(C) Finally we run exact *DPS-SMEPC* according to the prospect of evaluating the impact of the filtering rules described in Sections 4.2 and 4.3. It comes that we successively run *DPS-SMEPC*:

(1) while only using the *Strong Dominance* rule: then, ST(1) denotes the maximal number of states which have been generated this way for a given time pair (i, j) and $T(1)$ denotes related CPU time (in seconds);

(2) while using the *Strong Dominance* rule as well *Logical Filtering* rules: then, ST(2) denotes the maximal number of states which have been generated this way for a given time pair (i, j) and $T(2)$ denotes related CPU time (in seconds);

(3) while using the *Strong Dominance* rule as well *Logical Filtering* rules and *Upper/Lower Bound* based filtering rule, implemented while using the best value (according to Tab. 2) $W\_G(100)$: then, ST(3) denotes the maximal number of states which have been generated this way for a given time pair (i, j) and $T(3)$ denotes related CPU time (in seconds). ST(0) means the maximal number of possibly existing states for a given (i, j).

Results. Results related to (A) and (B) are summarized in Table 4. Results related to (C) are summarized in Table 5.

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. The introduction of the lower/upper bound filtering device has a very strong impact when the quality of the upper bound is good, as it is the case with the $W\_G(100)$ values of Table 4. Still, we keep on handling a large amount of states as soon as M and N become large. As for the greedy algorithm, it happens that in some cases, it applies excessively waiting strategies, and thus fails into getting good results, even through randomization. We should now try to make the process become more robust by applying an approximation DPS scheme involving a smaller number of states. It will be the purpose of a fore coming study.

TABLE 4. Values $G\text{-}Gap, W\text{-}Gap(x)$.

Inst.	G_G (%)	G_T (s)	$W\_G(20)$ (%)	$W\_T(20)$ (s)	$W\_G(50)$ (%)	$W\_T(50)$ (s)	$W\_G(100)$ (%)	$W\_T(100)$ (s)
1	2.2	0.1	2.2	0.1	0	0.1	0	0.2
2	33	0.3	31.5	0.4	10.2	0.6	0	1.0
3	5.2	0.4	5.2	0.4	0	0.7	0	1.2
4	31.2	0.2	30.5	0.3	19.9	0.4	0	0.7
5	0	0.1	0	0.1	0	0.1	0	0.2
6	12.1	0.3	12.1	0.3	21.6	0.5	0	0.9
7	23.1	0.2	10.8	0.3	0	0.5	0	0.8
8	19.6	0.1	19.6	0.2	15.9	0.3	0	0.4
9	2.5	0.2	0.8	0.2	0.8	0.3	0	0.4
10	44.3	0.3	42.3	0.4	37.1	0.6	0	1.1
11	6.9	0.4	6.9	0.6	6.9	1.1	6.1	2.0
12	11.3	0.6	10.6	0.8	9.2	1.3	6.4	2.2
13	47.9	0.7	42.6	1.0	35.1	1.7	9.6	2.8
14	6.4	0.3	6.4	0.4	5.7	0.7	2.7	1.2
15	17.7	0.8	19.0	1.0	14.8	1.6	1.3	2.7

TABLE 5. Values $\mathrm{ST}(0), \mathrm{ST}(1), \mathrm{ST}(2), \mathrm{ST}(3)$.

Inst.	ST(3)	$T(3)$ (s)	ST(2)	$T(2)$ (s)	ST(1)	$T(1)$ (s)	ST(0)
1	28	0.1	421	0.2	974	0.4	576 000
2	1350	2.1	16 658	49.2	33 424	103.3	705 600
3	916	1.7	13 476	22.0	30 023	50.01	718 848
4	121	0.2	10 989	11.5	38 261	40.1	2 416 128
5	312	0.2	48 501	181.4	97 751	373.6	876 096
6	3014	4.0	13 608	27.3	29 650	63.1	1 614 720
7	35	0.1	6013	10.0	13 141	22.8	199 920
8	101	0.1	887	0.7	4573	3.8	291 840
9	403	0.3	1718	1.1	3854	3.3	291 600
10	49	0.1	28 876	140.5	59 568	299.5	1 559 520
11	9675	66.7	118 795	1227.7	238 753	247.5	3 136 000
12	1988	11.7	16 028	135.7	36 244	292.3	1 206 120
13	10 329	121.9	31 810	492.9	74 802	1059.6	1 137 240
14	10 260	19.1	11 589	32.2	31 347	84.4	602 112
15	11 202	70.6	28 073	243.8	57 645	529.9	2 903 040

6. CONCLUSION

We have been presenting here a dynamic programming scheme in order to solve a scheduling problem which requires synchronizing tasks performed by vehicles with an energy production process. Many issues remain to be addressed:

- Extending our approach to several vehicles.
- Dealing with uncertainties related to H_2 production.
- Casting the routing issue into the decision process.

Most of all, we plan adapting the previously described DPS scheme in order to make it work in a more robust way and involve a smaller number of states.

Acknowledgements. This work was supported by the French National Research Agency in the Framework of the LabEx IMobS3 and the Région Auvergne-Rhone-Alpes.

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. *Disc. 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: Proceedings of the 17th 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] A. Burke, Batteries and ultracapacitors for electric, hybrid, and fuel cell vehicles. *Proc. IEEE* **95** (2007) 806–820.
[7] C.C. Chan, The state of the art of electric, hybrid, and fuel cell vehicles. *Proc. IEEE* **95** (2007) 704–718.
[8] P. Chretienne and A. Quilliot, Homogeneous non idling schedules of unit-time jobs on identical parallel machines. *Disc. Appl. Math.* **161** (2013) 1586–1597.
[9] P. Chretienne and A. Quilliot, A polynomial algorithm for the homogeneous non idling scheduling problem of unit-time independent jobs on identical parallel machines. *Disc. Appl. Math.* **243** (2018) 132–139.
[10] 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.
[11] E. Demaine, M. Ghodsi and M. Taghi Hajiaghayi, Scheduling to minimize gaps and power consumption. In: SPAA (2007) 46–54.
[12] M. Drexl, Synchronization in vehicle routing – a survey of VRPs with multiple synchronization constraints. *Transp. Sci.* **46** (2012) 297–316.
[13] S. Erdogan and E. Miller-Hooks, A green vehicle routing problem. *Transp. Res. Part E: Logist. Transp. Rev.* **109** (2012) 100–114.
[14] 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.
[15] C. Grimes, O. Varghese and S. Ranjan, Light, Water, Hydrogen: The Solargeneration of Hydrogen by Water Photoelectrolysis. Springer-Verlag US (2008).
[16] S. Irani and K. Pruhs, Algorithms for power management. *SIGACT News* **36** (2003) 63–76.
[17] 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.
[18] C. Koç, O. Jabali, J. Mendoza and G. Laporte, The electric vehicle routing problem with shared charging stations. *Int. Trans. Oper. Res.* **26** (2019) 1211–1243.
[19] Y. Kuo, Using simulated annealing to minimize fuel consumption for the time-dependent vehicle routing problem. *Comput. Ind. Eng.* **59** (2010) 157–165.
[20] A. Lajunen, Energy consumption and cost-benefit analysis of hybrid and electric city buses. *Transp. Res. Part C: Emerg. Technol.* **38** (2014) 1–15.
[21] S. Licht, Thermochemical and Thermal/Photo Hybrid Solar Water Splitting. Springer New York, New York, NY (2008).
[22] 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.
[23] L. Lozano and A.L. Medaglia, On an exact method for the constrained shortest path problem. *Comput. Oper. Res.* **40** (2013) 378–384.
[24] 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. *Int. J. Prod. Res.* **52** (2014) 3922–3939.
[25] 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.
[26] H. Mustapha, C. Desdouits, R. Giroudeau and C. Le Pape, Production scheduling with a piecewise-linear energy cost function (2016) 1–8.
[27] 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.
[28] M. Raylan, S. Matos, Y. Frota and L. Satoru Ochi, Green vehicle routing and scheduling problem with split delivery. *Electron. Notes Disc. Math.* **69** (2018) 13–20. Joint, EURO/ALIO, International Conference 2018 on Applied Combinatorial Optimization (EURO/ALIO 2018).
[29] M. Sachenbacher, M. Leucker, A. Artmeier and J. Haselmayr, Efficient energy-optimal routing for electric vehicles. In: AAAI (2011).
[30] M. Schneider, A. Stenger and D. Goeke, The electric vehicle-routing problem with time windows and recharging stations. *Transp. Sci.* **48** (2014) 500–520.
[31] R. Waraich, M. Galus, C. Dobler, M. Balmer, G. Andersson and K. Axhausen, Plug-in hybrid electric vehicles and smart grid: investigations based on a micro simulation. *Transp. Res. Part C: Emerg. Technol.* **28** (2014) 74–86.