Proceedings of the 18th Conference on Computer Science and Intelligence Systems pp. 667–676
DOI: 10.15439/2023F6717
ISSN 2300-5963 ACSIS, Vol. 35

Algorithmic Handling of Time Expanded Networks

Alain Quilliot and Jose-Luis Figueroa
LIMOS Lab.
UCA, CNRS and EMSE
Clermont-Ferrand, France
Email: alain.quilliot@uca.fr

Hélène Toussaint
and Annegret Wagler
LIMOS Lab.
Clermont-Ferrand, France

***Abstract*—Time Expanded Networks, built by considering the nodes of a base network over some time space, are powerful tools for the formulation of problems involving synchronization mechanisms. Those mechanisms may for instance be related to the interaction between resource production and consumption or between routing and scheduling. Still, in most cases, deriving algorithms from those formulations is difficult, due to both the size of resulting network structure and the fact that reducing this size through rounding techniques tends to induce uncontrolled error propagation. We address here this algorithmic issue, while proposing a generic decomposition scheme which works by first skipping the temporal dimension of the problem and next expanding resulting projected solution into a full solution of the problem set on the time expanded network.**

I. Introduction

ONE derives a *Time Expanded Network* (TE-Network) $N^{\mathbf{TIME}}$ (see [11]) from a base network $N = (X, A)$ and a time space **TIME**, by considering all the copies (x, t) of the nodes x of N at the different instants t of **TIME**. Then, an arc of $N^{\mathbf{TIME}}$ is either an arc $\big((x,t),(y,t+\delta)\big)$ which corresponds to the time required to traverse an arc (x, y) of N while starting from x at time t, or an arc $\big((x,t),(x,t')\big)$ with $t < t'$, which expresses some kind of standby in x from time t to time t'. It may happen that the traversal time δ depends on t. Note that the time space **TIME** may be either discrete or continuous.

Time Expanded Networks are powerful modeling tools for problems involving synchronization, between for instance resource production and consumption or between routing and scheduling. They are also well-fitted to deal with the time-dependence of a network. They were introduced by Ford and Fulkerson (see [11]) in order to cast such problems into the network flow framework. Some time later, concerns raised by time dependence and synchronization issues motivated the notion of *dynamic network* (see [2], [16]). They next gave rise at the beginning of the 80's to *flow over time* models, where flow values are trajectories, that means functions from a time space **TIME** onto real or integer numbers. Those functions may be subject to constraints like continuity or differentiability, and so the *flow over time* framework is well-fitted to the management of problems involving gas or power production and distribution. Years 1990/2000 also registered applications of these notions to evacuation planning (see [9]). At this time, authors adapted standard algorithms (min-cost flow and max-flow algorithms) and brought insights about the link between TE-Networks and the *flow over time* models (see [13]). They addressed complexity issues and stated some *polynomial approximation scheme* (PTAS) results. In the years 2010, authors came back to the original TE-Network framework. They did it with the purpose of handling multi-commodity flow models (see [1]) like those which may derive from transportation (see [5]) and industrial scheduling problems (see [18], [3]). They tried to take advantage of the improvement of both computers and *Mixed Integer Linear Programming* (MILP) libraries in order to directly implement some transportation models on those libraries (see [6], [17], [21]). They coped with the size issue by trying to adapt standard column generation and branch and cut techniques (see [7]), but faced difficulties as soon as the size of the time space increased. All those contributions suggested that the *Time Expanded Network* might be very efficient to deal with scheduling/routing problems involving synchronization (see [15], [14]) requirement or time dependence features (see [20]) .

Actually, though the TE-Network framework is good for modeling, it often fails in providing efficient algorithms. The fact is not only that the size of the TE-Network $N^{\mathbf{TIME}}$ increases very fast with the size of the time space **TIME**, but also that controlling this size through rounding techniques induces strong error propagation. So our purpose here is to bypass those difficulties by applying a *Project/Expand* decomposition scheme. We first search (*Project* step) for a good projected solution on the base graph N, and next (*Expand* step) turn this projected solution into a full TE-Network solution. We started addressing the *Project* step in a former contribution (see [12]), while setting a *Projected* model provided with *Extended No-Subtour* constraints enhancing its ability to yield full feasible solutions. We are now going to first reinforce this *Projected* model with constraints which guaranty the feasibility of the *Expand* process, and next derive from resulting projected solution a formal bi-level setting of related *Expand* problem. We shall deal with this problem in both an exact way and a heuristic way, by introducing a flexible active sub-network of $N^{\mathbf{TIME}}$ and tuning this sub-network until getting a full TE-Network solution. Though our approach is generic, we shall refer here, for the sake of understanding and with the purpose of testing, to a 2-commodity flow model related to the management of an *item balancing* process (see [8] and [17]).

The paper is organized as follows. In Section 2, we first present our reference TE-Network model, related to some *item balancing* problem, and next the projected model which derives from this TE-Network model. In Section 3 we set the *Expand* problem, characterize its feasibility and reinforce the projected model with constraints which ensure the feasibility of resulting *Expand* problem. In Section 4 we propose a bi-level formulation of this *Expand* problem and in Section 5 we present an exact MILP model together with heuristic algorithms which deal with the *Expand* problem. We conclude by providing some numerical experiments and discussing some open questions.

II. A TE-Network Model for the Item Relocation Problem

We refer here, for the sake of understanding, to a specific *Item Balancing Problem* (IBP). So, we consider here a transit network $N = (X, A)$, together with a $Depot$ node (see Figure 1). Every arc is provided with a time value $T_{x,y}$ and a cost value $C_{x,y}$. Also:

- We set $\mathbf{T} = (T_{(x,y)}, (x,y) \in A)$ and $\mathbf{C} = (C_{(x,y)}, (x,y) \in A)$. For any path π from $x \in X$ to $y \in X$, we denote by $L^T(\pi)$ its length in the sense of $\mathbf{T}$. We do the same with $\mathbf{C}$. For any pair of nodes (x, y) we denote by $D^T(x, y)$ the shortest path distance from x to y in the sense of $\mathbf{T}$, and by $D^C(x, y)$ the shortest path distance from x to y in the sense of $\mathbf{C}$.
- Let U be some subset of X. We set $\partial_N^-(U) = \{(x,y) \in A$ such that $x \notin U, y \in U\}$, $\partial_N^+(U) = \{(x,y) \in A$ such that $x \in U, y \notin U\}$, $\partial_N(U) = \partial_N^-(U) \cup \partial_N^+(U)$, and $A(U) = \{(x,y) \in A$ such that $x \in U, y \in U\}$. Clearly, $\partial_N(U)$ means the arcs which allow entering and getting out of U. We simplify these notations in case U is a singleton $\{x\}$ by writing $\partial_N^-(x)$, $\partial_N^+(x)$ and $\partial_N(x)$, instead of $\partial_N^-(\{x\})$, $\partial_N^+(\{x\})$ and $\partial_N(\{x\})$, respectively. Also, we denote by $U \setminus V$ the difference of the sets U and V (i.e., the set $\{u \in U$ such that $u \notin V\}$).

Then our IBP: *Item Balancing Problem* comes as follows: Items are located inside the network and must be relocated, within a *time horizon* $\{0, 1, \ldots, T_{max}\}$, by a fleet of identical carriers with *capacity* Cap. We are provided with an integral *balance* vector $\mathbf{b} = (b_x, x \in X)$ such that $\sum_{x \in X} b_x = 0$: $b_x > 0$ means that x is in *excess* and that b_x items must leave x; $b_x < 0$ means that x is in *deficit* and that b_x items must arrive to x. The *Item Balancing Problem* (IBP) consists in scheduling those transfers, while minimizing a hybrid cost $\alpha \cdot c_1 + \beta \cdot c_2 + \gamma \cdot c_3$, where c_1 is the number of active carriers, c_2 is their running cost in the sense of $\mathbf{C}$, c_3 is the time spent by items while moving inside the carriers, and α, β, γ are scaling coefficients. An important feature of the problem is that we allow *preemption*, which means that the carriers may exchange items, making synchronization become an issue.

Example 1 Consider the network $N = (X, A)$ of Figure 1. It shows two carrier routes $\Gamma^1 = (Depot, v, x, y, Depot)$ and $\Gamma^2 = (Depot, w, x, z, Depot)$ linked together by a transfer from Γ^1 to Γ^2 at node x. We check $c_1 = 2$, $c_2 = 10$, and $c_3 = 32$.

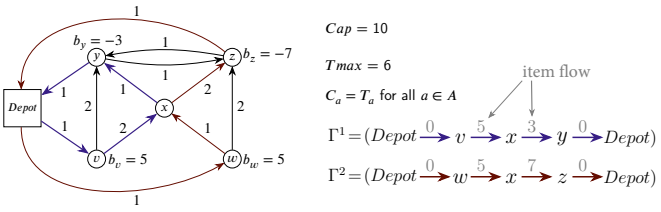

Fig. 1. The transit network $N = (X, A)$ used in Example 1

A. A Time-expanded 2-commodity Flow Model for the IBP Problem

In order to cast this IBP problem into the TE-Network framework (see [8]), we first derive from the network $N = (X, A)$ its *time expansion* $N^{Tmax} = (X^{Tmax}, A^{Tmax})$ according to $Tmax$. The node set X^{Tmax} is the set of all pairs (x, t), $x \in X$, $t \in \{0, 1, \ldots, Tmax\}$, augmented with two distinguished nodes: $source$ and $sink$. The arcs $a \in A^{Tmax}$, together with their *carrier cost* $\hat{C}_a$, and their *item cost* $\hat{I}_a$, come as follows:

- *input-arcs* $a = \big(source, (x, 0)\big)$, $x \in X$, with $\hat{I}_a = 0$ and $\hat{C}_a = 0$;
- *output-arcs* $a = \big((x, Tmax), sink\big)$, $x \in X$, with $\hat{I}_a = \hat{C}_a = 0$;
- *waiting-arcs* $a = \big((x,t), (x, t+1)\big)$, $x \in X$, $t \in \{0, \ldots, Tmax - 1\}$, with $\hat{I}_a = \hat{C}_a = 0$;
- *active-arcs* $a = \big((x,t), (y, t + T_{(x,y)})\big)$, $(x,y) \in A$, $t \in \{0, \ldots, Tmax - T_{(x,y)}\}$, with $\hat{I}_a = \gamma \cdot T_{(x,y)}$ and $\hat{C}_a = \beta \cdot C_{(x,y)}$;
- *backward-arc* $a = (sink, source)$, with $\hat{I}_a = 0$ and $\hat{C}_a = \alpha$.

In order to formalize our IBP problem as a 2-commodity flow model on this network $N^{Tmax} = (X^{Tmax}, A^{Tmax})$, we introduce integral 2 flow vectors $\mathbf{H}$ and $\mathbf{h}$, both indexed on the arc set of N^{Tmax}. The first one is going to describe the way the carriers move inside the transit network: for any active-arc $a = \big((x,t), (y, t + T_{(x,y)})\big)$, H_a will mean the number of carriers which traverse the arcs (x, y) of the transit network between time t and time $t + T_{(x,y)}$. The second one will describe the moves of the items: for any active-arc $a = \big((x,t), (y, t + T_{(x,y)})\big)$, H_a will mean the number of items which traverse the arcs (x, y) of the transit network between time t and time $t + T_{(x,y)}$. The value H_a on the *backward-arc* will provide us with the number of carriers involved into the process. The values H_a and h_a on a *waiting-arc* $a = \big((x,t), (x, t+1)\big)$ are going to respectively provide us with the number of carriers and items waiting on node x between time t and time $t + 1$. By proceeding this way, we get:

TE-Network IBP Model. *Compute two nonnegative integral A^{Tmax}-indexed vectors* $\mathbf{H}$ *and* $\mathbf{h}$ *(for carriers and items, respectively) such that:*

- **H** *and* **h** *satisfy flow conservation at any node of* N^{Tmax}*;* (E1)
- *for any active-arc* $a = \big((x,t),(y,t+T_{(x,y)})\big)$*:* $h_a \leq Cap \cdot H_a$ *;* (E2)
- *for any input-arc* $a = \big(source,(x,0)\big)$*,* $x \neq Depot$*:* $H_a = 0$*;* $h_a = \max(b_x, 0)$*;* (E3)
- *for any output-arc* $a = \big((y,Tmax), sink\big)$*,* $y \neq Depot$*:* $H_a = 0$*;* $h_a = \max(-b_y, 0)$*;* (E4)
- *the global cost* $Cost(H,h) = \sum_{a \in A^{Tmax}} \big(H_a \cdot \hat{C}_a + h_a \cdot \hat{I}_a\big)$ *is minimized.*

Constraints (E1) express the circulation of carriers and items. Constraints (E2) mean that any item move is supported by some carrier. Constraints (E3) and (E4) characterize initial and final states: carriers start and end at $Depot$, while any node ends as *neutral*.

Example 2: Fig. 2 shows the TEN $N^{Tmax} = (X^{Tmax}, A^{Tmax})$ deriving from network $N = (X, A)$ of Fig. 1 and $T_{max} = 6$. It turns the solution of Example 1 into a full solution (**H**, **h**). Tour Γ^1 gives rise to a path $\{source, (Depot,0), (v,1), (x,3), (y,4), (Depot,5), (Depot,6), sink\}$. Tour Γ^2 gives rise to a path $\{source, (Depot,0), (w,1), (x,2), (x,3), (z,5), (Depot,6), sink\}$. The value of **h** on the arc $(x,2),(x,3)$ shows the way the 5 items transported by carrier 2 wait on node x before splitting themselves along the arc (x,y) and (x,z).

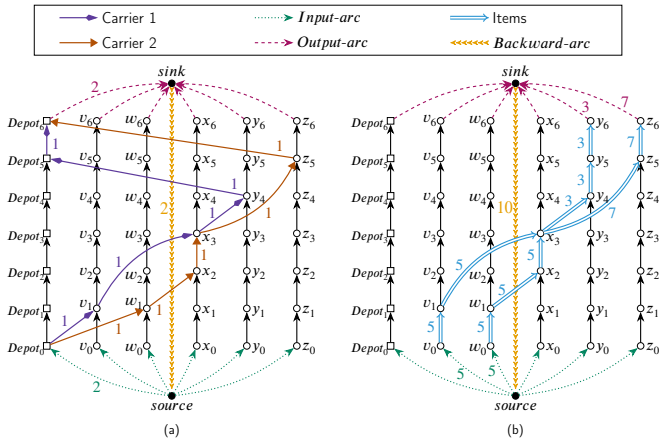

Fig. 2. The routes and schedules as a TE-Network 2-commodity flow: (a) Carrier flow vector **H**. (b) Item flow vector **h**.

As it is formulated, we understand that this **TE-Network IBP Model** is going to be difficult to handle. Its size grows fast with value T_{max}. We might try to control this size by rounding the values t. For instance, instead of measuring the time in minutes, we could try to round it to 15 minutes packages. But if traversing an arc requires 3 minutes, then rounding will mean either canceling this traversal time, or multiplying it by 5. In both case, we guess that the propagation of resulting errors will be difficult to manage. So we are going to implement a *Project and Expand* scheme, which means that we are going to first skip the temporal dimension of our problem (*Project* step) and restrict ourselves to the identification of the arcs followed by the carriers and the items, and next perform an *Expand* step in order to schedule those arcs and get a full solution of our problem.

B. The Projected IBP Model

Given a feasible solution (**H**, **h**) of above **TE-Network IBP** Model. We define the *projection* **F** of **H** on the network N by setting, for any arc (x,y) of N:

$$F_{(x,y)} = \sum_{t=0}^{Tmax} H_{(x,t),(y,t+T_{(x,y)})}.$$

We define the same way the *projection* **f** of **h**. Clearly, the meaning of those projected vectors is that we want to simplify our problem while skipping its temporal dimension. They are going to provide us with a kind of signature of the routes followed respectively by the carriers and the items on the transit network, without taking care neither of the order according to which they run along the arcs of this transit network nor of the timestamps telling when they do it. Of course, we expect that computing those projected vectors **F** and **f** will be easier than computing **F** and **f**. In order to perform this computation, we must characterize those vectors.

We see that **F** and **f** must be such that:

- **F** satisfies flow conservation at any vertex of X; (E5.1)
- for any node x of N: $\sum_{a \in \partial_N^+(x)} f_a - \sum_{a \in \partial_N^-(x)} f_a = b_x$; (E5.2)
- for any arc a of N: $f_a \leq Cap \cdot F_a$; (E6)

According to this, carrier riding cost c_2 and item riding time c_3 come as follows:

- carrier riding cost $c_2 = \beta \cdot \big(\sum_{a \in A} C_a \cdot F_a\big)$; (E7.1)
- items riding time $c_3 = \gamma \cdot \big(\sum_{a \in A} T_a \cdot f_a\big)$. (E7.2)

Still, this formulation is not enough in order to efficiently characterize **F** and **f**. First, (E5.1)-(E7.2) fail in estimating the carrier number $c_1 = H_{(sink,\,source)}$. In order to make our projected model provide a good estimation of the carrier number, we proceed as follows:

Approximating the carrier number: We first notice as in [12] that the quantity $\sum_{a \in A} T_a \cdot F_a$ means the global time that carriers spend running inside N, waiting times being excluded. Since the whole process must last no more than $Tmax$ time units, it requires at least $\left\lceil \frac{\left(\sum_{a \in A} T_a \cdot F_a\right)}{Tmax} \right\rceil$ carriers. Thus, (**F**, **f**) should minimize the *projected cost*:

$$PCost(\mathbf{F},\mathbf{f}) = \alpha \cdot \frac{\left(\sum_{a \in A} T_a \cdot F_a\right)}{Tmax} + \beta \cdot \big(\sum_{a \in A} C_a \cdot F_a\big) + \gamma \cdot \big(\sum_{a \in A} T_a \cdot f_a\big).$$

Next, we notice that constraints (E5.1) and (E5.2) do not forbid subtours. In order to forbid subtours, we proceed in an augmented way:

The *Extended No-Subtour* Constraint: Given a subset $U \subset X \setminus \{Depot\}$. The time that carriers spend moving at the border or inside U, is equal to $\sum_{a \in \partial_N(U) \cup A(U)} T_a \cdot F_a$. For each carrier q, this time cannot exceed $Tmax$. If Q denotes the number of carriers involved into an IBP solution, then

we see that $Q \cdot Tmax \geq \sum_{a \in \partial_N(U) \cup A(U)} T_a \cdot F_a$. Since $\sum_{a \in \partial_N^-(U)} F_a \geq Q$, we deduce that the following *Extended No-Subtour* inequality should hold:

$$Tmax \cdot \left(\sum_{a \in \partial_N^-(U)} F_a \right) \geq \sum_{a \in \partial_N(U) \cup A(U)} T_a \cdot F_a. \quad \text{(E8)}$$

This leads us to set the following *projected* problem about the search for **F** and **f**:

PIBP: *Projected Item Balancing* Problem

{*Compute on the network* $N = (X, A)$ *two nonnegative integral vectors* A*-indexed* **F** *and* **f** *such that:*

- **F** *satisfies flow conservation at any node of* X*;* (E5.1)
- *for any node* $x \in X$, $\sum_{a \in \partial_N^-(x)} f_a - \sum_{a \in \partial_N^+(x)} f_a = b_x$*;* (E5.2)
- *for any arc* $a \in A$, $f_a \leq Cap \cdot F_a$*;* (E6)
- *for any* $U \subseteq X \setminus \{Depot\}$: $Tmax \cdot \left(\sum_{a \in \partial_N^-(U)} F_a \right) \geq \sum_{a \in \partial_N(U) \cup A(U)} T_a \cdot F_a$; (E8)
- *Minimize* $PCost$(**F**, **f**)$= \alpha \cdot \frac{\sum_{a \in A} T_a \cdot F_a}{Tmax} + \beta \cdot \left(\sum_{a \in A} C_a \cdot F_a \right) + \gamma \cdot \left(\sum_{a \in A} T_a \cdot f_a \right)$. (E9)}

We proved in [12], that the *Extended No-Subtour* constraints may be separated in polynomial time. A consequence is that this projected problem may be efficiently solved while using a MILP library and implementing a Branch and Cut process.

III. The Expand Issue

So we come now to the *Expand* step. That means that we suppose that we have been computing (**F**, **f**) as above and that we want to derive (**H**, **h**) in a satisfactory way. First, we notice that, in many cases, there will not exist (**H**, **h**) whose projection is equal to (**F**, **f**). Figure 3 shows that a PIBP solution (**F**, **f**) may not be the projection of any feasible IBP TE-Network solution (**H**, **h**): The carrier must follow the route $(Depot, y, x, z, y, Depot)$ and will never be able to transport that way any item from z to x.

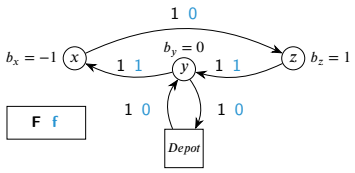

Fig. 3. A solution of PIBP which cannot be expanded.

So we must put some flexibility while setting our *Expand* issue. Namely, we should set it as follows: How can we derive from (**F**, **f**) a good full IBP solution (**H**, **h**)? In a more generic way, how can we efficiently deal with a TE-Network model while applying the following resolution scheme?

***Project/Expand* Decomposition Scheme.**

1) Solve a projected version of the problem which skips the temporal dimension.
2) Turn (*expand*) resulting solution (**F**, **f**) into a *"good"* solution (**H**, **h**) of the original problem, while restricting ourselves to a reduced representation of related TE-Network.

One feels that there are several ways to interpret above *expand* part of the process. Intuition would suggest to search for (**H**, **h**) such that (**F**, **f**) is the projection of (**H**, **h**). Such a setting is NP-Hard (see [8]). But the true problem is that both above example and numerical experiments show that such a setting is too strong and most often does not admit any feasible solution. So what we decide in the present case is to formalize step 2 while only requiring **f** to be the projection of **h**. So we set the **Expand** problem as follows:

***Expand* Problem** EXPAND(**f**)**.** {*Compute a feasible IBP solution* (**H**, **h**) *such that:*

- *The projection of* **h** *on the transit network* N *is equal to* **f** ;
- *The cost value* $Cost$(**H**, **h**) *is the smallest possible.*}

This **Expand** problem is difficult. One may check that EXPAND(**f**) contains both the **TSP**: *Traveling Salesman Problem* and the standard **Pick up and Delivery** problem. It is far more general, since we may by no way identify the arcs which support f with requests and must take care of the precedence relations implicitly related to those arcs. Above example of Figure 3 shows that part of our problem consists in determining in which order the arcs supporting **f** must be visited.

Before going further towards the design of algorithmic solution for this *Expand* problem, we must address the issue related to its feasibility: Can we characterize the conditions which will make this **Expand** problem admit a feasible solution? The answer is positive, and checking it is going to provide us with a reinforcement of the *Projected* model of Section II, that means with a way to compute (**F**, **f**) which will guaranty this feasibility.

Checking the Feasibility of EXPAND(f): Enhancing the PIBP model.

We just told that a too strong setting of the *Expand* issue could lead to unfeasible situations. So a natural question comes about the feasibility of above EXPAND(**f** problem. In order to deal with it, let us introduce the following notion of *feasible path*. A *feasible path* is a path that an item may follow while moving from an *excess* node to a *deficit* one, taking into account that the carrier which transports it must be able to move from the *Depot* node to the start-node of this path and next from the end-node of this path until the *Depot* node, within the time horizon $\{0, 1, \ldots, T_{max}\}$. Clearly, flow vector **f** should be such that all items may follow feasible paths. We formalize this by saying that:

- A *feasible path* of N is any path π from $x \in X^+$ to $y \in X^-$ whose length $L^T(\pi)$ in the sense of the time is such that: $D^T(Depot, x) + L^T(\pi) + D^T(y, Depot) \leq Tmax$. We denote by $\mathbf{f}^\pi$ related $\{0, 1\}$ flow vector and by Π^{FP} the set of feasible paths.
- Vector **f** is *feasible-path-decomposable* iff it can be written: $\mathbf{f} = \sum_{\pi \in \Pi^{FP}} \lambda_\pi \, \mathbf{f}^\pi$, with $\lambda_\pi \geq 0$.

Since any item in node $x \in X^+$ must be transported to some vertex $y \in X^-$ along a feasible path, we get that **f** must feasible-path-decomposable. But checking that **f** is

feasible-path-decomposable is just a matter of solving a rational linear program, and characterizing the feasibility of this linear program may be done by using Duality Theory. More precisely, we define a *Path Feasibility* vector as any Π^{FP}-indexed vector $\mathbf{w} = (w_\pi, \pi \in \Pi^{FP})$ such that:

For any feasible path π, we have $\sum_{a\in\pi} w_a \geq 0$.

This allows to state:

Theorem 1: EXPAND(**f**) *is feasible iff, for any Path Feasibility vector* **w**, *we have:* $\sum_{a\in A} f_a \cdot w_a \geq 0$.

Sketch of the Proof: Linear Programming Duality makes that (E10) holds iff **f** is feasible-path-decomposable. This property is clearly necessary in order to allow the existence of **H** and **h**. We get sufficiency by considering such a decomposition of **f** and assigning to any item its own feasible path and a carrier which transports it along this feasible path. We explicitly build this way flow vectors **H** and **h**. **EndProof**

So we reinforce our **PIBP** model by imposing vector **f** to be feasible-path-decomposable:

For any *Path Feasibility* vector **w**: $\sum_{a\in A} f_a \cdot w_a \geq 0$. (E10)

Theorem 2: *(E10) can be separated in polynomial time.*

Sketch of the Proof: Every time we are provided with a flow vector **f** (rational or integral) we search for a *feasible-path* decomposition of **f**. This means solving some linear program with respect to some current collection Λ of feasible paths. In case of failure, then we apply duality and generate another feasible path, until we succeed or we get a *Path Feasibility* vector which contradicts (E10). The time-polynomiality of this separation process derives from the time-polynomiality of Rational Linear Programming. **EndProof**

It comes that the **PIBP** model reinforced by (E10) may be efficiently handled through Branch-and-Cut.

IV. A Bi-Level Decomposition Scheme for the Expand Problem

The purpose of this section is to show the way the **Expand** problem man be decomposed in a way which will allow us in next section V to implement both exact and heuristic algorithms. The main idea behind this decomposition scheme is that the quality of final solution (**H**, **h**) is mostly determined by the way items are routed. Intuitively, that means that we should first expand flow vector **f**, and next try to *cover* it by a carrier flow **H** according to the following 2-step approach:

1st step: *Expand* item flow **f** into an item flow **h** on the TE-Network N^{Tmax}, while relying on a specific $Split(N, \mathbf{f})$ network. This network is going to split those arcs and nodes of network N which are supporting **f** (in practice, it will mean few arcs and nodes) according to item packages likely to be carried by the same carriers. This will allow us to make appear the feasible paths followed by the items when moving from an excess node to a deficit nodes. Providing *ad hoc* time values to the nodes exploded this way will yield item flow vector **h**.

2nd step: Once **h** has been fixed, extend **h** into a good (best) solution IBP (**H**, **h**), while solving a Min-Cost Flow problem on the *active* part of a specific $Carrier(N, \mathbf{f})$ network. Network $Carrier(N, \mathbf{f})$ is going to extend above mentioned network $Split(N, \mathbf{f})$ in order to make appear the possible moves performed by the carriers. Related *active* part will be made of the arcs which are consistent with the time values related to vector **h**.

Linking 1st step and 2nd step: The quality of the resulting solution deeply depends not only on the route followed by the items, but also on the time values of the vertices related to **h** in N^{Tmax}. We shall delay, as long as possible, the instantiation of those time values while relying on a flexibility device which will take the form of a collection Λ of arcs common to both $Split(N, \mathbf{f})$ and $Carrier(N, \mathbf{f})$. This device will identify the moves that carriers and items are allowed to perform when switching from an arc of N supporting items to another one. This arc collection Λ will become the master object of a bi-level *Split/Carrier* decomposition scheme.

A. The Networks $Split(N, \mathbf{f})$ and $Carrier(N, \mathbf{f})$

The Network $Split(N, \mathbf{f})$.

The purpose of the network $Split(N, \mathbf{f})$ is to help us in describing the way items traverse any vertex x of the network N, and so the trajectories followed by the items when moving from the excess nodes to the deficit ones. So we build it while relying on the arcs of network N which support non null **f** and exploding the nodes involved into those arcs in order to make appear the routes followed by the items.

Nodes of $Split(N, \mathbf{f})$: With any arc $a = (x, y) \in A$ such that $f_a \geq 1$, we associate $\lceil \frac{f_a}{Cap} \rceil$ *copy-arcs* $a^m, m = 1, \ldots, \lceil \frac{f_a}{Cap} \rceil$, with respective origin $p = (x, a, m, +)$ and respective destination $q = (y, a, m, -)$. We denote by $Copy(a)$ the set of those arcs $a^m, m = 1, \ldots, \lceil \frac{f_a}{Cap} \rceil$. At the same time we create those copy-arcs, we also create *copy-nodes* $p = (x, a, m, +)$ and $q = (y, a, m, -)$, which respectively correspond to the carriers who leave x with a non-null load and to the carriers who arrive into y with a non-null load. Resulting node set X^* becomes the node set of $Split(N, \mathbf{f})$. We denote by $Copy(A)$ the set of all those copy-arcs. For any node $p = (y, a, m, \varepsilon)$ of X^*, we set $x(p) = y$ and $\varepsilon(p) = \varepsilon$. Also, for any node y of N, we set:

- $X^*(y) = \{p \in X^* \text{ such that } x(p) = y\}$;
- $X^*Plus(y) = \{p \in X^* \text{ such that } x(p) = y \text{ and } \varepsilon(p) = +\}$;
- $X^*Minus(y) = \{p \in X^* \text{ such that } x(p) = y \text{ and } \varepsilon(p) = -\}$.

Arcs of $Split(N, \mathbf{f})$: We complete the arc collection $\{a^m, a = (x, y), \text{ such that } f_a \geq 1, m = 1, \ldots, \lceil \frac{f_a}{Cap} \rceil\}$ by *middle-arcs* which, for any node x of N, connect copy-nodes $(x, a, m, -)$ (with a arriving into x), $m = 1, \ldots, \lceil \frac{f_a}{Cap} \rceil$ to copy-nodes $(x, a', m', +)$ (with a' starting from x) $m' = 1, \ldots, \lceil \frac{f_a}{Cap} \rceil$. We denote by $Middle$ the set of all middle-arcs created that way, and, for any node x of N, we denote by $Middle(x)$ the set of all middle-arcs u whose origin may be written $(x, a, m, -)$. $Middle(x)$ defines a complete bipartite

graph on the nodes of $X^*(x)$. For any vertex $p = (x, a, m, +)$, we denote by $MiddleIn(p)$ the set of middle-arcs u with destination p, and for any vertex $q = (x, a, m, -)$, we denote by $MiddleOut(q)$ the set of middle-arcs u with origin q. We also set:

- $CopyIn(y) =$ $\{a \in Copy(A)$ with destination in $X^*Minus(y)\}$;
- $CopyOut(y) =$ $\{a \in Copy(A)$ with origin in $X^*Plus(y)\}$.

We denote by $Split(N, \mathbf{f})$ the resulting network, that one may check to be acyclic. This construction is illustrated in Figure 4.

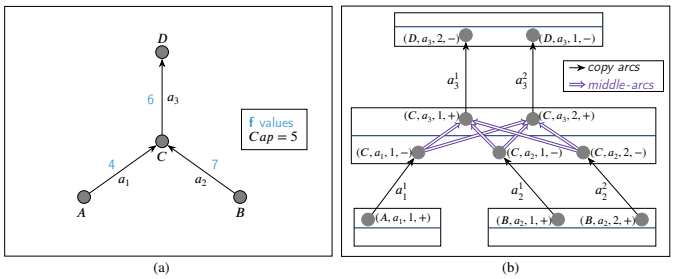

Fig. 4. Building the $Split(N, \mathbf{f})$ Network

The network $Carrier(N, \mathbf{f})$.
This network is going to help us in computing the carrier routes, in such a way that those carrier routes *cover* the item routes. So, it will contain exactly the same nodes as the network $Split(N, \mathbf{f})$, but two additional nodes *source* and *sink*. Its arcs will express all the ways a carrier may move from an arc supporting items to another one while following a shortest path in N i the sense of vector $\mathbf{C}$. Depending on the time values assigned to its nodes, those arcs will be allowed or not to support non null carriers flow values. It comes that this network will behave as a flexible reduced version of N^{Tmax}.

Nodes of $Carrier(N, \mathbf{f})$: They are all nodes of $Split(N, \mathbf{f})$, augmented with two nodes $source$ and $sink$. We denote by $V(Carrier)$ the node set of $Carrier(N, \mathbf{f})$.

Arcs of $Carrier(N, \mathbf{f})$: They are the arcs of $Split(N, \mathbf{f})$ augmented with:

- one *back* arc $u = (sink, source)$, provided with cost $Q_u = \alpha$;
- any *in* arc $u = \big(source, p = (x, a, m, +)\big)$, $p \in X^*$, provided with a cost Q_u equal to the cost of a time-minimal path (in the sense of cost matrix $\mathbf{C}$) from $Depot$ to x;
- any *out* arc $u = (p = (x, a, m, -), sink)$, $p \in X^*$, provided with a cost Q_u equal to the cost of a time-minimal path from x to $Depot$;
- any *transverse* arc $u = \big(p = (x, a, m, -), q = (y, a', m', +)\big)$, with $p, q \in X^*$, provided with a cost Q_u equal to the cost of a time-minimal path from x to y.

We denote by $A(Carrier)$ the arc set of the network $Carrier(N, \mathbf{f})$.

B. The Split/Carrier Decomposition Scheme

The way item flow values f_a, $a \in \partial_N^-(x)$ arriving into a node x distribute themselves into values $f_{a'}$, $a' \in \partial_N^+(x)$ leaving x while moving through x, is going to be described by two integral vectors $\mathbf{z} = \big(z_u,\ u = \big((x, a, m, -), (x, a', m', +)\big) \in Middle\big)$, and $\mathbf{z}^* = \big(z^*_{a^m},\ a^m \in Copy(A)\big)$. Those two vectors will provide us with an *expanded* vector $\mathbf{h}$ once we assign time values t_p to the nodes p of the $Split(N, \mathbf{f})$. On the other side, the routes followed by the carriers are going to be described by a $\{0, 1\}$-valued flow vector $\mathbf{Z}$ defined on the arcs of the network $Carrier(N, \mathbf{f})$. In order to link those vectors $\mathbf{z}$, $\mathbf{z}^*$, $\mathbf{t}$, and $\mathbf{Z}$ together we need a *mediator* object. This *mediator* object is going to be here a collection Λ of arcs in $A(Carrier)$, providing us with the *transverse* arcs and the *middle* arcs supporting the item and carrier moves. It will induce the following constraints:

- on the time vector $\mathbf{t} = (t_p, p \in V(Carrier))$: for every arc (p, q) in Λ, $t_q \geq t_p + D^T(x(p), x(q))$, which means that any carrier or item move along an arc of the network $Carrier(N, \mathbf{f})$ requires a time at least equal to the length of the path in N which is related to such an arc;
- on the $\mathbf{z}$ vector: for every *middle-arc* $u = (p, q)$ which is not in Λ, $z_u = 0$;
- on the flow vector $\mathbf{Z}$ representing the carrier routes on the network $Carrier(N, \mathbf{f})$: for every *transverse* arc $u = (p, q)$ which is not in Λ, $Z_u = 0$.

More precisely, once Λ has been determined, we get that vector $\mathbf{t}$ must satisfy the following linear constraint system CTIME(Λ), with underlying totally unimodular constraint matrix.

CTIME(Λ) constraint system on $\mathbf{t} = (t_p,\ p \in X^* \cup \{source, sink\})$:

- $t_{source} = 0$; (E11)
- $t_{sink} \leq Tmax$; (E12)
- *For any in arc* $u = \big(source, p = (x, a, m, +)\big)$, $p \in X^*$: $t_p \geq D^T(Depot, x)$; (E13)
- *For any out arc* $u = \big(p = (x, a, m, +), sink\big)$, $p \in X^*$: $Tmax \geq D^T(x, Depot)$; (E13-1)
- *For any copy arc* $u = \big(p = (x, a, m, +), q = (y, a, m, -)\big)$, $a = (x, y) \in A$ *such that* $f_a \neq 0$: $t_q \geq t_p + D^T(x, y)$; (E13-2)
- *For any arc* (p, q) *in* Λ: $t_q \geq t_p + D^T(x(p), x(q))$. (E13-3)

As for vectors ($\mathbf{z}$, $\mathbf{z}^*$), they must express the way items move from any arc a support of $\mathbf{f}$ in N and with destination x to another one with origin x. More precisely:

- $\mathbf{z}$ is going to express, for any node x of network N, the way items arriving along a copy-arc a^m into a copy-node $q = (y, a, m, -)$ are going to distribute themselves among the copy-arcs $a'^{m'}$ leaving x. Clearly, the feasibility of this distribution process will impose Λ to contain enough arcs of $Middle(x)$.

- $\mathbf{z}^*$ is going to express, for any arc a of network N, the way item flow values f_a distribute themselves among the copy-arcs related to a.

This yields the following linear constraint system Split(N, **f**, Λ), with underlying totally unimodular matrix and whose feasibility depends on Λ:

Split(N, f, Λ) constraint system on vectors z, z*:

- *For any copy-arc* a^m: $z^*_{a^m} \leq Cap.$ (E14)
- *For any copy-node* $q = (y, a, m, -)$: $z^*_{a^m} \leq \sum_{u \in MiddleOut(q)} z_u.$ (E15)
- *For any copy-node* $p = (x, a, m, +)$: $z^*_{a^m} \geq \sum_{u \in MiddleIn(p)} z_u.$ (E16)
- *For any node* x *of* N: $\sum_{v \in CopyIn(x)} z^*_u = \sum_{u \in Middle(x)} z_u + \max(-b_x, 0).$ (E17)
- *For any node* x *of* N: $\sum_{v \in CopyOut(x)} z^*_u = \sum_{u \in Middle(x)} z_u + \max(b_x, 0).$ (E18)
- *For any middle-arc* $u \notin \Lambda$: $z_u = 0.$ (E19)

Finally, the flow vector **Z** with indexation on the arcs of $Carrier(N, \mathbf{f})$ and which is going to provide us with the arcs and paths followed by the carriers, should be a solution of the following Min-Cost Flow model Carrier(N, **f**, Λ).

Carrier(N, f, Λ) constraint system on vector **Z**:

- **Z** *satisfies flow conservation.* (E20)
- *For any arc* u *in* $Copy(A)$: $Z_u = 1.$ (E21)
- *For any transverse or middle arc* $u \notin \Lambda$: $Z_u = 0.$ (E22)
- *Cost value* $\sum_{u \in A(Covering)} Q_u Z_u$ *is minimal.* (E23)

We may now reformulate the **Expand** Problem as the following bilevel ([19]) setting.

***Split/Carrier* Reformulation of the Expand Problem.** *Compute* $\Lambda \subseteq A(Carrier)$ *restricted to middle and transverse arcs, such that:*

- CTIME(Λ) *admits a feasible solution;*
- Split(N, **f**, Λ) *admits a feasible solution* (**z**, **z***)*;*
- *The optimal value* $\sum_{u \in A(Carrier)} Q_u Z_u$ *of* Carrier(N, **f**, Λ) *is minimal.*

It may happens that above *Split/Carrier* model does not admit any feasible solution, while the **Expand** Problem is feasible. In such a case, we say that **f** is *Split/Carrier* {inconsistent. Figure 5 displays an example of such a situation:path A, B, C, D is not a feasible path and so **f** must be decomposed into 2 feasible paths, while the Split(N, **f**, Λ) model would allow only 1 feasible path. However, numerical experiments will show that it happens very scarcely.

Still, we may state:

Theorem 4.1: *Any feasible solution of above Split/Carrier model is a feasible solution of EXPAND(f), with same cost.*

Sketch of the Proof: It comes through an algorithmic construction of (**H**, **h**) from arc collection Λ. Time vector **t** obtained through resolution of CTIME(Λ) allows us to embed the nodes of the network $Carrier(N, \mathbf{f})$ into the time expanded network $N^{Tmax} = (X^{Tmax}, A^{Tmax})$. Then we derive **h** from a solution (**z**, **z***) of Split(N, **f**, Λ) and we derive **H** from a solution **Z** of Carrier(N, **f**, Λ). **EndProof**

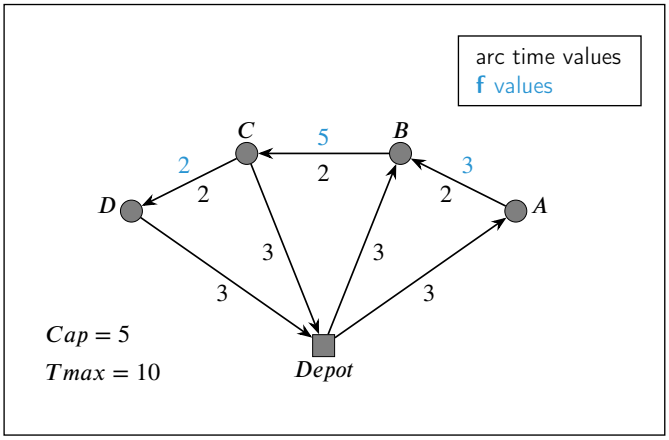

Fig. 5. *Split/Carrier* Inconsistent Flow Vector **f**)

V. Algorithmic Handling the *Split/Carrier* Decomposition Scheme

The *Split/Carrier* decomposition scheme involves, as its master object, a collection Λ of arcs of the $Carrier(N, \mathbf{f})$ network. Constraints of Split(N, **f**, Λ) are transportation constraints set on a bipartite graph. Carrier(N, **f**, Λ) is a Min-Cost Flow model while CTIME(Λ) is about the computation of the largest path in an acyclic graph ([4]). It comes that those 3 sub-problems may be viewed as *easy* and that we may rely either on Carrier(N, **f**, Λ) duality or on the largest paths which arise from the resolution of CTIME(Λ) in order to drive the master object Λ. We are going to describe here the ways we implemented this idea.

A. An Exact MILP Resolution

We turn previous *Split/Carrier* decomposition scheme into an Expand(N, **f**) MILP model. We do it by considering vectors **Z**, **z**, **z*** and **t** as in above Section IV, introducing an additional vector $\mathbf{X}^\Lambda$ with indexation on the *middle* and *transverse* arcs of the $Carrier(N, \mathbf{f})$ network, and merging the programs CTIME(Λ), Split(N, **f**, Λ and Carrier(N, **f**, Λ) into a unique one, according to the following modifications:

- (E13-3) is replaced by: For any *transverse* or *middle* arc $u = (p, q)$: $t_q \geq t_p + D^T(x(p), x(q))$.
- (E19) is replaced by: For any*middle* arc u: $z_u \leq X^\Lambda_u$.
- (E20) is replaced by: For any *transverse* or *middle* arc u: $Z_u \leq X^\Lambda_u$.

B. A Greedy Dual_Carrier Algorithm

This algorithm works in a greedy way, while making increase the arc collection Λ. We first initialize Λ in such a way that the constraint system Split(N, **f**, Λ) admits a solution. We do it by removing constraint (E19), and defining Λ as the set of *middle* arcs which support non null **z** values. Then, at every iteration of the main loop of the greedy process, we are provided with some current arc collection Λ, and we use a feasible solution **t** of CTIME(Λ) in order to set a version of the min-cost flow model which replace (E22) by:

For any (*transverse* or *middle*) arc (p, q) such that:
$t_q < t_p + D^T(x(p), x(q))$, we have $Z_u = 0$. (E32-1)
This may be summarized as follows:

- 1st step: Solve the Split(N, **f**, Λ) while considering only constraints (E14)-(E18). Derive vectors $\mathbf{z}^*$ and **z** and ℓ, and initialize Λ with the arcs of the network $Carrier(N, \mathbf{f})$ which support $\mathbf{z}^*$ and **z**.
- 2nd step: While Not $Stop$ do:
  1) Get a feasible solution **t** of CTIME(Λ;
  2) Solve the Carrier(N, **f**, Λ) Min-Cost Flow model modified by replacing (E22) by (E22-1);
  3) Search for an arc (p, q) whose insertion into Λ makes related dual solution become infeasible and maintains the feasibility of CTIME(Λ). If $Success(Search)$ then insert this arc into Λ else $Stop$.

C. A Greedy Path_Concatenate Algorithm

Once again, we make the collection Λ increase, while using the CTIME(Λ) constraint system in order to deduce, at any iteration of the main loop, which arc (p, q), with $p = (x_1, a_1, m_1, -)$ and $q = (x_2, a_2, m_2, +)$ has to be inserted in Λ.

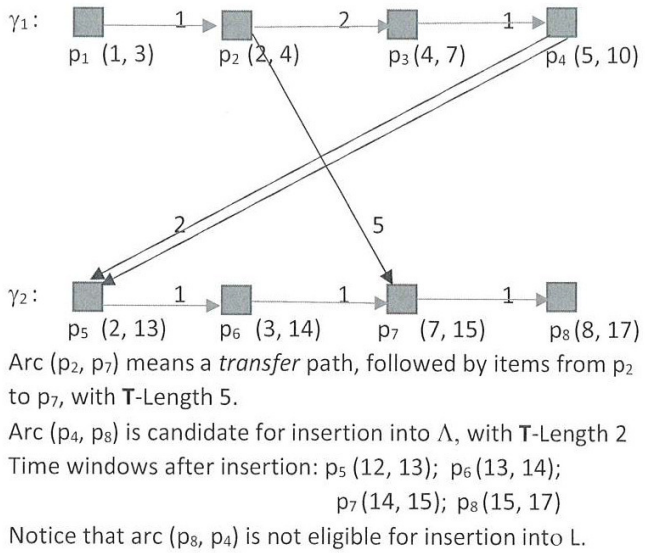

Fig. 6. *Concatenating 2 carrier paths γ_1 and γ_2)*

More specifically, we first solve Split(N,**f**, Λ) obtained while removing constraint (E19) and initialize collection Λ with the *middle* arcs which support non null **z** values. Then we compute some feasible solution **t** of CTIME(Λ), set Λ(**t**) $= \{(p, q) \in A(Carrier) : t_q \geq t_p + D^T(x(p, x(q)))\}$, and solve the Carrier($N$, **f**, Λ(**t**)) Min-Cost Flow problem. This provides us with an initial solution **H**, as well as with a collection Γ of paths that the carriers follows between the first time they load some item and the last time they unload an item.

Next we proceed iteratively, while trying at every iteration to concatenate two paths γ_1 and γ_2 of Γ into a unique one as in ([10]), so that a same carrier can follow those two paths and go back to $Depot$ without violating the time horizon constraint (see Figure 6). We do it, while relying on constraint propagation, in such a way that resulting path is the shortest possible in the **C** sense.

More precisely, we proceed in two steps:

- **Initialization**: Solve Split(N, **f**, Λ) with $\Lambda = \emptyset$ and derive vectors **z**, $\mathbf{z}^*$ which will remain unchanged during all the process. Initialize Λ with the arcs of $Carrier(N, \mathbf{f})$ which correspond to non-zero **z** values. Compute some feasible solution **t** of CTIME(Λ), set Λ(**t**) $= \{(p, q) \in A(Carrier) : t_q \geq t_p + D^T(x(p, x(q)))\}$, and solve resulting Carrier(N, **f**, Λ(**t**)) Min-Cost Flow problem. Derive an initial solution **H**, as well as a collection Γ of paths that the carriers follow between the first time they load some item and the last time they unload an item. Propagate current constraints of CTIME(Λ) and get, for any node $p = (z, a, m, \varepsilon) \in X^*$ a time window $[Inf_p, Sup_p]$, with $Sup_p \geq T_{max} - D^T(x, Depot)$ and $Inf_p \leq D^T(Depot, x)$.
- **Main Loop**: It works as follows:
  1) Denote by $StartCarrier$ the set of all nodes $p_1 = (x_1, a_1, m_1, +)$ which are the start nodes of the paths of current collection Γ and by $EndCarrier$ the set of all nodes $p_2 = (x_2, a_2, m_2, -)$ which are the end node of the paths of current collection Γ.
  2) Select $p_2 = (x_2, a_2, m_2, -) \in EndCarrier$ and $p_1 = (x_1, a_1, m_1, +) \in StartCarrier$ such that $Inf_{p_2} + T_{max} - Sup_{p_1}$ does not exceed T_{max}, which also means $Inf_{p_2} \leq Sup_{p_1}$, and such that $Sup_{p_1} - Inf_{p_2}$ is largest possible. If $Fail(Select)$ then $Stop$ else keep on with 3) and 4).
  3) Insert arc (p_2, p_1) into Λ. Update the time windows induced by the constraint system CTIME(Λ) and compute some feasible solution **t** of CTIME(Λ).
  4) Set Λ(**t**) $= \{(p, q) \in A(Carrier) : t_q \geq t_p + D^T(x(p, x(q)))\}$ and solve the Carrier(N, **f**, Λ(**t**)) Min-Cost Flow problem. Update accordingly the best solution **H** ever found.

  Stop occurs at instruction 2), when it is not possible to find a new arc (p_1, p_2) to insert into collection Λ.

D. Numerical Tests

According to this, we perform several numerical experiments, whose purpose is 3-sided:

1) Evaluating the error induced by the projection step, that means the gap between the value of the projected **PIBP** model and the value of the full TE-Network model.
2) Evaluating the error induced by the 2-step *Project and Expand* process, with respect to a theoretical value which would be obtained by solving in an exact way the full TE-Network **IBP** model.
3) Evaluating the ability of both heuristics *Dual_Carrier* and *Path_Concatenate* to compute in a short time an

efficient solution, with respect to the value obtained from application of a MILP library to the exact model Expand(N, **f**).

Technical Context: We run those experiments on a computer with a 2.3GHz Intel Core i5 processor and 16GB RAM, while using the C++ language (compiled with *Apple Clang 10*) and the CPLEX12.10 MILP library.

Instances: No standardized benchmarks exist for the generic IBP. So we built instances as follows: the node set X is a set of n points inside a 100×100 grid, the set of arcs A consists of m arcs generated randomly, the time matrix **T**= $(T_{(x,y)}, (x,y) \in A)$ corresponds to the rounded Euclidean Distance and the cost matrix **C**= $(C_a, a \in A)$ to the Manhattan Distance. Each node x but $Depot$ is assigned to a b_x value in $\{-10, \ldots, 10\}$, the capacity Cap is chosen in $\{2, 5, 10, 20\}$, and the time horizon limit $Tmax$ is a product $\lambda \cdot (\max_{(x,y)\in A} T_{(x,y)})$ when choosing $\lambda \in \{4, 5, 6, 8, 9\}$. The scaling coefficients α, β, γ are chosen in such a way that the values of cost components $\alpha \cdot$ *number of carriers*, $\beta \cdot$ *carrier riding cost* and $\gamma \cdot$ *items riding time* become comparable.

Outputs. Table I involves 15 instances and displays:

- Values n, m, respectively the number of vertices and arcs of the base network N.
- Values Cap, $Tmax$, respectively the carrier capacity and the time horizon value.
- Values **G1**, **V1**, respectively the optimal value of the projected model and related carrier number.
- Values **G**,**V**, respectively an upper bound value of the full TE-Network IRP model computed by the CPLEX library in no more than 1 h (this relatively short time limit is due to the fact that we use a personal computer), and related carrier number.

Table II involves the same instances as table I and displays:

- Values **GDC**,**VDC**, **TDC**, respectively the value computed by the *Dual_Carrier* Algorithm, related carrier number and related running time (in seconds).
- Values **GPC**,**VPC**, **TPC**, respectively the value computed by the *Path_Concatenate* Algorithm, related carrier number and related running time (in seconds).
- Values **GL**,**VL**, respectively the optimal value computed of the exact MILP Expand(N, **f**) model (computed through CPLEX Library) and related carrier number.
- Missing values are indicated by a hyphen symbol -, and correspond to PIBP solutions **f** for which the Expand(N, **f**) MILP is unfeasible.

Comments: The *Path_Concatenate* heuristic finds feasible IBP solutions for most instances but only 1, which happens not to be *Split/Carrier* consistent. Also, by comparing values **GDC** and **GPC**, **TDC** and **TPC**, **VDC** and **VPC**, we see that most of the time the solutions found by the *Path_Concatenate* heuristic have lower costs, involve fewer vehicles, and have required lower running times than the solutions computed by the *Dual_Carrier* Algorithm. Finally, comparing values **G**, **GL**, **V** and **VL** shows us that the *Project/Expand* decomposition scheme yields a very good approximation of the optimal value of the TE-Network IBP model. By the way, we notice an instance (instance 12) which really puts the MILP solver in trouble. Examining this instance makes appear that it is very tight, and admits few efficient feasible solutions. Moreover, related vehicle number coefficient α is rather large. So we feel that it was difficult for the MILP solver to turn rational solution into feasible integral solutions.

TABLE I
BEHAVIOR OF THE *Dual_Covering* AND *Path_Concatenate* ALGORITHMS.

Id	n	m	Cap	$Tmax$	**G1**	**V1**	**G**	**V**
1	20	78	2	324	2110.85	3	2633.00	4
2	20	65	5	400	1196.10	3	1282.7	3
3	20	77	10	440	854.83	2	1123.25	2
4	20	75	5	400	1105.00	2	1269.3	3
5	20	81	10	440	887.9	2	1005.05	2
6	50	163	2	460	15561.30	17	17043.00	20
7	50	155	5	390	4326.10	7	5023.7	9
8	50	149	10	440	7966.03	6	8820.5	8
9	50	146	20	436	1840.17	4	2670.95	6
10	50	168	20	436	2169.5	5	2750.10	6
11	100	363	2	336	17179.00	22	19483.00	27
12	100	236	5	516	4826.24	8	31881.35	86
13	100	289	10	432	6091.24	4	8450.00	9
14	100	296	5	516	4320.5	6	12029.4	20
15	100	308	10	432	6340.4	5	9875.4	12

TABLE II
BEHAVIOR OF THE *Dual_Carrier* AND *Path_Concatenate* ALGORITHMS.

Id	**GDC**	**TDC**	**VDC**	**GPC**	**TPC**	**VPC**	**GL**	**VL**
1	3097.00	0.17	5	3097.00	0.003	5	2633.00	4
2	1289.50	0.12	3	1289.50	0.100	3	1289.50	3
3	1493.25	0.44	4	1495.25	0.041	3	1468.35	3
4	1302.4	0.52	3	1269.3	0.25	3	1269.3	3
5	1212.9	077	3	1074.6	0.38	2	1005.05	2
6	20952.00	780.61	35	18435.00	19.007	24	17043.00	20
7	5328.9	122.6	10	5023.7	47.9	9	5023.7	9
8	-	-	-	-	-	-	-	-
9	2859.0	158.2	8	2711.4	54.8	7	2670.95	6
10 **G**	2900.8.0	226.0	7	2784.3	82.0	6	2750.10	6
11	23469.00	303.52	35	21623.00	9.303	32	20086.00	28
12	9331.00	382.99	24	6571.75	20.803	14	6436.0	14
13	10906.2	188.6	12	7548.00	65.3	7	7459.3	7
14	6224.7	405.6	10	5468.5	208.5	8	5029.4	8
15	9408.5	252.4	12	7977.0	109.0	10	7900.3	10

VI. CONCLUSION

We just presented a *Project/Expand* decomposition scheme for the handling of 2-commodity flow problems set on TE-

Networks and involving a coupling constraint. We focused here on the *Expand* issue, and proposed approaches based on the implicit management of the TE-Network.

Still, some issues remain open. One of them is the time dependency, when the state of the network evolves over time. Another one is about dynamicity and robustness since routing decisions are usually taken in a dynamic way and must cope with some uncertainty.

ACKNOWLEDGMENT

Present work was funded by French ANR: National Agency for Research, Labex IMOBS3, and PGMO Program.

REFERENCES

[1] R. K. Ahuja, T. L. Magnanti, J. B. Orlin, M. R.Reddy, *Applications of network optimization*, Chapter 1 of Network Models, Handbook of Operation Research and Management Science 7, pp. 1-83, 1995. doi.org/10.1016/S0927-0507(05)80118-5

[2] J. Aronson, *A survey of dynamic network flows*, Ann. Oper. Res. 20, pp. 1-66, 1989. doi.org/10.1007/BF02216922

[3] C. Artigues, E. Hébrard, A. Quilliot, H. Toussaint, *Models and algorithms for natural disaster evacuation problems*, Proc. 2019 FEDCSIS WCO Conf., p 143-146, 2019. doi.org/10.15439/2019F90

[4] R. Bellman, *On a routing problem*, Quarterly of Applied Mathematics, 16, p 87-90, 1958.

[5] F. Bendali, J. Mailfert, E. Mole-Kamga, A. Quilliot, H. Toussaint, *Pipelining dynamic programming process in order to synchronize energy production and consumption*, Proc. 2020 FEDCSIS WCO Conf., p 303-306, 2020. doi.org/10.15439/978-83-955416-7-4.

[6] F. Bendali, J. Mailfert,and A. Quilliot, *Flots entiers et multi-flots fractionnaires couplés par une contrainte de capacité*, Investigacion Operativa, 9, 2001. DOI : 10.1051/ro:2006003

[7] S. Bsaybes, A. Quilliot, A. Wagler, *Fleet management for autonomous vehicles using flows in time-expanded networks*, TOP, Springer Verlag 27 (2), pp. 288-311, 2019. DOI: 10.1007/s11750-019-00506-4

[8] D. Chemla, F. Meunier,*Bike sharing systems: the static rebalancing problem*, Discrete Optimization 10 (2), p 120-146, 2013. doi.org/10.1016/j.disopt.2012.11.005

[9] A. O.Fleischer, M. Skutella, *Quickest flows over time*, SIAM Journal of Computing 36 (6), p 1600-1630, 2007. doi.org/10.1137/S0097539703427215

[10] S. Fidanova, O. Roeva, M. Ganzha, *Ant colony optimization algorithm for fuzzy transport modelling*, Proc. 2020 FEDCSIS WCO Conference, p 237-240, 2020. doi.org/10.15439/978-83-955416-7-4

[11] R. Ford and D. Fulkerson, *Flows in networks*, Princeton University Press, 1962.

[12] J. L.Gonzalez, M. Baiou,A. Quilliot, H. Toussaint, A. Wagler,*Branch and cut for a two commodity flow relocation model with time constraints*, Combinatorial Optimization. ISCO 2022. LNCS 13526. Springer, Cham. 2022. doi.org/10.1007/978-3-031-18530-4-2

[13] M. S.Hall and S. Hippler, *Multi-commodity flows over time*, Theoretical Computer Sciences, p 58-84, 2007.

[14] N. Kyngas, K. Nurmi, *The extended shift minimization personnel task scheduling problem*, Annals of Computer Sciences and Information Systems 26, p 65-74, 2021. doi.org/10.15439/978-83-959183-9-1

[15] K. Kishkin, D. Arnaudov, V. Todorov, S. Fidanova, *Multicriterial evaluation and optimization of an algorithm for charging energy storage elements*, Annals of Computer Sciences and Information Systems 26, p 61-64, 2021. doi.org/10.15439/978-83-959183-9-1

[16] W. B.Powell and P. Jaillet, *Stochastic and dynamic networks and network routing*, Handbook Operations Research, North Holland, 1995.

[17] F. T.Raviv and M. Tzur, *Static repositionning in a bike sharing system: models and solution approaches*, EURO Journal of Transportation and Logistics 2, p 187-229, 2013. DOI 10.1007/s13676-012-0017-6

[18] J. Schuijbroek, R. C. Hampshire, W. Van Hoeve, *Inventory rebalancing and vehicle routing in bike sharing systems*, EJOR 257 (3), (2017). doi.org/10.1016/j.ejor.2016.08.029

[19] K. Stoilova, T. Stoilov, *Bi-level optimization application for urban traffic management*, Proc. 2020 FEDCSIS WCO Conf., p 327-336, 2020. doi.org/10.15439/978-83-949419-5-6

[20] S. Varone, D. Schindl, C. Beffa, *Flexible job shop scheduling problemm with sequance-dependent transportation constraints and setup times*, Annals of Computer Sciences and Information Systems 26, p 97-102, 2021. doi.org/10.15439/978-83-959183-9-1

[21] Q. P. Zheng, A. Arulselvan, *Discrete time dynamic traffic assignment models and solution algorithms for managing lanes*, Journal of Global Optimization 51, p 47-68, 2011. doi.org/10.1007/s10898-010-9618-5