

A Service Network Design Problem for Freight Transportation in Port Cities

Massimo Di Francesco(✉), Dennis Incollu, Claudia Porcu, and Simone Zanda

Department of Mathematics and Computer Science, University of Cagliari,
09124 Cagliari, Italy
{mdifrance,simone.zanda}@unica.it
d.incollu@gmail.com,claudia.porcu92@gmail.com

Abstract. This paper investigates a service network design problem, which is motivated by the case of freight transportation in a port city. We describe the novel features of this problem, which are based on the possible (but still unexploited) knowledge on the composition of the disaggregated loads in containers and semitrailers entering the port. In this problem, the transportation requests of these loads are highly customized and have different delivery costs. We must determine the paths of vehicles and loads, which result in costs paid by carriers and customers, as well as external costs for the city itself. The resulting network design problem is faced from the holistic viewpoint of a possible mobility manager, which must minimize the overall system costs. We present a mixed integer linear programming model (MILP) for this problem. Since it is very difficult to solve by standard MILP solvers, we present a Tabu Search algorithm exploiting the specific problem features. The computational experiments show to what extent this problem can be tackled by a general purpose mixed-integer programming solver and the Tabu Search algorithm.

Keywords: Network design problems · City logistics · Tabu Search

1 Introduction

Freight transportation is a fundamental element enabling several economic and social activities in urban areas (for example, stores and offices need to be supplied, items need to be delivered at home and citizens must get rid of garbage). It is a crucial link between suppliers and customers and involves several stakeholders (e.g. carriers, depots, shippers, receivers, etc.). However, freight transportation generates nuisances in urban areas. Freight vehicles compete with vehicles transporting people for the capacities of streets and contribute to congestion and related issues (e.g. emissions, pollution and noise). Clearly, these nuisances impact the lives of people in cities, the productivity of the firms in urban areas,

Supported by *Fondazione di Sardegna* through the project *Algorithms for Approximation with Applications*.

© Springer Nature Switzerland AG 2020
O. Gervasi et al. (Eds.): ICCSA 2020, LNCS 12255, pp. 262–277, 2020.
https://doi.org/10.1007/978-3-030-58820-5_21

as well as the efficiency of the same freight transportation services. These problems are expected to continue growing, owing to logistics practices based on low inventories and time delivery, particularly in the domain of business-to-customer electronic commerce. Moreover, the problems are amplified by the increasing urbanization: in 2014, 54% of the world population lived in urban areas this percentage is expected to increase up to 66% until 2050 [1].

Freight transportation in urban areas has been subject to a significant amount of research [2,3]. The general objective is to decrease the impact of freight transportation on city living conditions without penalizing social and economic activities. More specifically, the goal is to reduce the number and the dimensions of freight vehicles and increase the efficiency of their movements. The key idea is to stop considering each shipment, firm, and vehicle individually and start viewing individual stakeholders as parts of an integrated logistics system, which must be optimized. The term "City Logistics" has been adopted to denote the need for an optimized consolidation of loads of different shippers in the vehicles of different carriers for the coordination of the freight transportation activities within the city [4]. City Logistics plans must be arranged by local/national authorities to make decisions and evaluate policies in order to regulate urban freight flows. Sometimes these plans are made by urban mobility managers, who need to be supported by optimization-based methods.

The research on these methods is based on the representation of cities as holistic systems, where consolidation and coordination activities are performed at facilities organized into a hierarchical two-echelon structure and two facility types: major terminals, which are located at the city limits, and satellite facilities, which are smaller redistribution points in the city where the goods are transshipped into smaller vehicles for the final distribution [5]. Particular vehicle fleets are dedicated to each echelon. However, this network topology does not apply to port cities, in which the port represents a very major terminal located in the city centre [6]. This paper investigates this variant, which often arises in island territories.

The port represents the main entrance for freight entering and leaving islands. In these areas freight transportation is necessarily intermodal (or multimodal), as goods are moved from their origin to their destination by a sequence of at least two transportation modes [7]. To improve the efficiency of the whole distribution process, loads (e.g. pallets) are consolidated in containers (or semi-trailers) for long-haul maritime transportation, while taking advantage of the efficiency of local pick-up and delivery operations by truck for the last-mile distribution. Carriers may either provide a customized service, where the overall container is dedicated exclusively to a particular customer (full-load service), or operate on the basis of consolidation (less-than-truckload service), where each container moves freight for different customers with possibly different origins and destinations.

In this problem the destinations of the loads in containers are supposed to be located in the city surrounding the port (i.e. if the destinations are out of the city, containers are moved out of the city and are not of interest for the urban

mobility manager (e.g. [8]); we also neglect the freight flows originating from the city, as most islands are import dominant). Containers cannot be opened in the port, because it aims to quickly remove containers from docks to create room for containers arriving shortly from the seaside and the landside. The destinations in the city cannot be typically joined by vehicles carrying containers, because they cannot enter many streets or this is disallowed by local regulations, to reduce congestion and pollutant emissions.

To deal with this problem, a special two-echelon distribution structure needs to be adopted. In the first echelon, containers are moved by vehicles from the port to intermediate facilities called satellites, where loads are unpacked from containers and packed into smaller and, possibly, environment-friendly vehicles (the activity of satellites is also called cross-docking [9]). In the second echelon, loads are moved from satellites to their destinations by the smaller vehicles. To our knowledge, a similar distribution scheme was investigated in studies on dry ports, which are facilities *directly connected to one or more seaports with high-capacity transport means, where customers can drop and pick up their standardized units as if directly at a seaport* [10]. Dry ports can act as special satellites to alleviate congestion in the hinterland of maritime harbours [11].

Typically containers are *a priori* assigned to satellites according to existing contracts regardless of the destinations of their loads. Clearly, this *modus operandi* is often inefficient, as "better" satellites can be selected if one knows the destinations of the loads in containers. For example, a mobility manager could determine the "average" destination of the loads in a container and select the closest satellite to this artificial destination. More important, the precise composition of the loads in each container can be derived from data on transportation requests, but it is still almost unexploited for the planning of freight transportation in maritime cities. Nevertheless, the potential of technological developments in data sharing represents viable perspectives to deal with such inefficiencies effectively.

This paper investigates the service network design problem for freight transportation in the second echelon of a port city. More precisely, we assume that containers are already assigned to satellites according to predefined (and, possibly, cost-effective) policies and focus on inbound containers filled with loads to be delivered within the city. The objective is to support a mobility manager in the construction of a transportation plan to serve the freight demand by the vehicles of several consolidation-based carriers, while operating at the same time in an efficient and profitable manner. This service network design problem aims to determine the routes of vehicles and the itineraries of each load from the satellite to the final destination.

This paper aims to introduce readers without a background in Operations Research (but research interests in ICT, urban planning and logistics) to explore its potential in approaching this challenging problem. First, the problem is mathematically described by a mixed integer programming model (MILP), in which the legs of paths of vehicles and loads are represented by boolean or integer decision variables. They must be determined, such that all customers are served

with their loads by the available vehicles and the overall system costs are minimized. These costs are described by a linear function of the decision variables and the objective is to minimize this function. The need to serve customers by the available vehicles is enforced by linear constraints in the decision variables. Therefore, the service network design problem is presented as a constrained optimization problem (or MILP model), which can be solved (i.e. to determine decision variables) by algorithms.

Several MILP solvers do incorporate exact optimization algorithms, which aim to determine the least-cost problem solution (in this problem the least-cost paths for vehicles and loads). However, service network design problems are very difficult to solve, because in the worst case the computational time increases exponentially with the size of the instance (these problems are said to be $\mathcal{NP} - hard$, which means that it is very unlikely to solve them in polynomial time). Therefore, the exact algorithms of MILP solvers are expected to solve only small problem instances of our service network design problem, i.e. they cannot be useful for mobility managers in dealing with City Logistics. Nevertheless, in this paper we investigate which instances of this new problem are in the grasp of a standard MILP solver.

In order to solve larger problem instances, we propose an algorithm which determines the paths of vehicles and loads, even if the overall system costs are not guaranteed to be minimized. This algorithm is a metaheuristic in the class of Tabu Search methods. It is based on a sequence of local changes to the paths of vehicles and loads and adopts specific memories (the so-called Tabu Lists), to avoid solutions already visited during the search. Two variants of Tabu Search are presented: in the first variant, the algorithm visits only the set of feasible solutions (i.e. the set of paths of vehicles and loads that satisfy all the constraints). In order to improve the solutions, we investigate an algorithmic variant in which the temporary visits of infeasible solutions is allowed and penalized to reestablish their feasibility (i.e. the ability to satisfy all the constraints). Some computational experiments are reported, to discuss the effectiveness of the MILP solver and that of the Tabu Search algorithms.

This paper is organized as follows. In Sect. 2 we describe this service network design problem and illustrate the MILP model for this problem. In Sect. 3 we describe the Tabu Search algorithm. In Sect. 4 we report the experimentation of the MILP solver and the Tabu Search algorithms. We list conclusions and research perspectives in Sect. 5.

2 Problem Setting and Modeling

Consider a fleet of inbound containers in a satellite. Each container is filled with loads (or pallets), which have different destinations (or customers) in the city. Each final customer is associated with a known demand of the loads packed in containers. The transportation service is highly customized, as different costs are paid by customers for the transportation of loads to each destination. As a result, each demand can be seen as a different commodity, which can be identified by its destination. A fleet of vehicles is available in the satellite to serve

the transportation requests. They are heterogeneous in terms of transportation capacity and routing cost per unitary distance. Moreover, we assume that the overall fleet of vehicles provides a sufficient transportation capacity to serve all the requests of customers.

In this problem setting, the routes of vehicles are supposed to be open, i.e. they are not required to return to the satellite after servicing the last customer in the route. Moreover, splitting is allowed for all destinations of the second tier, i.e. the loads requested by each customer can be provided by several vehicles.

The problem can be described by the following MILP formulation. Let s be a satellite, K the set of customers and V the set of vehicles. Consider a directed graph $G = (N, A)$, where the set N of nodes is defined as $\{s\} \cup K$ and the set A of arcs consists of all possible links from the satellite or a customer to any other customer: $A = \{(i, j) : i \in s \cup K, j \in K, i \neq j\}$. If vehicle $v \in V$ moves along arc $(i, j) \in A$, a routing cost $c^v(i, j)$ is paid. In the simplest setting, this cost is paid by the carrier, but it can also incorporate external costs paid by the city due to the movement of this vehicle. If vehicle $v \in V$ moves the loads with destination $k \in K$ from node $i \in N$ to node $j \in N$, a unitary transportation cost $f_k^v(i, j)$ is paid by customer $k \in K$ for this transportation service. Let d_k be the number of loads requested by customer $k \in K$ and u_v the maximum number of loads carried by vehicle $v \in V$.

The following quantities (or decision variables) must be determined:

$y_k^v(i, j)$**:** it represents the number of loads with destination $k \in K$ shipped along arc $(i, j) \in A$ by vehicle $v \in V$;
$x^v(i, j)$**:** it takes value 1 if vehicle $v \in V$ traverses arc $(i, j) \in A$, 0 otherwise.

According to the former notation, the following objective function is minimized: $z = \sum_{v \in V} \sum_{(i,j) \in A} \left(c^v(i, j) x^v(i, j) + \sum_{k \in K} f_k^v(i, j) y_k^v(i, j) \right)$. It accounts for the overall system costs from the viewpoint of mobility managers. The first term represents the costs generated by the movement of each vehicle along each traversed arc. It can incorporate not only the routing costs of carriers, but also the costs for the city (even if their quantification is beyond the scope of this paper). The second term represents the overall costs generated by each load traversing specific arcs by specific vehicles. It accounts for the costs of shippers, but it can also be used by mobility managers to penalize the transportation of specific commodities on specific arcs by specific vehicles.

Decision variables must satisfy the following constraints:

- All loads with destination $k \in K$ must leave the satellite; this constraint is enforced as follows: $\sum_{v \in V} \sum_{j \in N - s} y_k^v(s, j) = d_k, \ \forall k \in K$;
- Each destination $k \in K$ must receive its own pallets; this constraint is imposed as follows: $\sum_{v \in V} \sum_{i:(i,k) \in A} y_k^v(i, k) = d_k, \ \forall k \in K$;
- The loads with destination $k \in K$ traverse any customer $k' \in K$ visited before $k \in K$; this constraint is formulated as follows: $\sum_{i:(i,k') \in A} y_k^v(i, k') - \sum_{i:(k',i) \in A} y_k^v(k', i) = 0 \ , \ \forall v \in V \ , \ \forall k', k \in K, k' \neq k$;
- Each vehicle is used at most once; this constraint is formulated as follows: $\sum_{i \in K} x^v(s, i) \leq 1 \ , \ \forall v \in V$;

– If a vehicle joins a customer, it leaves and moves toward the next customer, or it terminates the service, if the last customer in the route has been served; this constraint is enforced as follows:
$\sum_{i:(i,k)\in A} x^v(i,k) \geq \sum_{l:(k,l)\in A} x^v(k,l)$, $\forall v \in V$, $\forall k \in K$;
– The total demand of the customers assigned to the route of vehicle $v \in V$ does not exceed its capacity u_v in each arc; this constraint is imposed as follows:
$\sum_{k\in K} y_k^v(i,j) \leq u_v x^v(i,j)$, $\forall v \in V$, $(i,j) \in A$.

A feasible solution for this problem consists in a collection of paths of vehicles and loads from the satellite to a subset of customers. The total demand of the customers in the path of a vehicle is not larger than the capacity of the vehicle.

To the best of our knowledge, this problem has not been addressed yet, even if some similar problems were investigated in the literature on routing. Tavakkoli-Moghaddam et al. [12] investigated a vehicle routing problems with splits and heterogeneous fleets, but they ignored the customer-dependent transportation costs. Cattaruzza et al. [14] investigated a multi-commodity vehicle routing problem without splits to minimize the number of vehicles. Archetti et al. [13] adopted a path-based model to investigate the routing costs of vehicles capable of carrying any set of commodities, but ignored transportation costs.

3 Tabu Search

Approximate solution techniques (or heuristics) have been broadly used for solving combinatorial optimization problems. The most popular ones are based on Local Search (LS) improvement techniques: starting from an initial feasible solution, it is progressively improved by applying a series of local modifications (or moves). At each iteration, LS switches to an improving feasible solution, such that the difference between the previous and the new solution is one move. The LS terminates when it encounters a local optimum with respect to the considered transformations, but this local optimum is often a solution of low or moderate quality. Tabu Search (TS) allows LS methods to overcome local optima: TS continues LS whenever a local optimum is visited by allowing non-improving moves and preventing cycling back to previously visited solutions by memories (or tabu lists), which record the recent history of the search. The basic elements of any LS or TS algorithm are the search space and its neighborhood structure. The search space is the space of all possible solutions that can be visited during the search. The neighborhood of the current solution is the set of solutions obtained by applying a single move to it [15].

In the following subsections we present a TS algorithm tailored to our service network design problem. For the sake of clarity, we first introduce the simplest variant, in which the search space is the set of feasible paths. It is called *Basic Tabu Search* and denoted by TS_B. Next, we present a variant in which the visit of infeasible solutions is temporarily allowed and penalized. It is called *Tabu Search with Penalties* and denoted by TS_P.

3.1 Basic Tabu Search

Several initial solutions are built by two ordered lists of pallets for each destination and vehicles available in the satellite. The first initial solution is determined as follows. Starting from the top of the list of pallets, all pallets of the first destination are assigned to the first vehicle, if it has sufficient capacity, else a part of the pallets are assigned to the second vehicle. Next, the pallets of the second destination are assigned to the first available vehicle in the list and so on. The other initial solutions are built by the same procedure after a random reordering of the two former lists.

For each initial solution, the following tabu search steps are performed. The search space is the set of feasible solutions to the problem. Therefore, each point in the search space corresponds to a set of loads carried in each segment of a route performed by a vehicle. Two types of non-tabu moves are considered:

***1-0 Exchange* (or *customer relocate*):** a part or the totality of the loads (or pallets) for a destination are moved at each iteration from the current position in a route. The pallets for the selected destination are inserted in the same route or in another route with sufficient residual capacity. We evaluate all possible *0-1 exchange* moves and select the move resulting in the minimum overall cost in the next iteration.

***1-1 Exchange* (or *customer exchange*):** all pallets for two destinations are swapped at each iteration from their current positions, which can be in the same route or in two different routes. The vehicles involved in this move are required to have sufficient residual capacity. We evaluate all possible *1-1 exchange* moves and select the move resulting in the minimum overall cost in the next iteration.

In Fig. 1 the caption (v_1, v_2, n_1, n_2, p) describes the *1-0 exchange* move of p pallets with destination n_2 from vehicle v_2 to vehicle v_1, which will serve n_2 after n_1. In the left side of the figure, the current solution is reported; in the right side, we show the solution after the local move. The following cases can occur:

case (a) One pallet for node D is moved to route v_1 and delivered after node A. Thus, in the LS one must remove arc (A, B) and add arcs (A, D) and (D, B) in route v_1. If all pallets for node D are removed from route v_2, in the LS one also needs to remove arcs (C, D) and (D, E) and add arc (C, E) in route v_2.

case (b) Two pallets for node E are moved to route v_1 and delivered after node A. Thus, in the LS one must remove arc (A, B) and add arcs (A, E) and (E, B) in route v_1. If all pallets for node E are removed from route v_2, in the LS one also needs to remove arcs (C, D) and (D, E) and add arc (C, E) in route v_2.

case (c) One pallet for node D is moved to route v_1 and delivered after node B. Thus, in the LS one must add arcs (B, D) at the end of route v_1. If all pallets for node D are removed from route v_2, in the LS one also needs to remove arcs (C, D) and (D, E) and add arc (C, E) in route v_2.

case (d) Two pallets for node E are moved to route v_1 and delivered after node B. Thus, in the LS one must add arcs (B, E) at the end of route v_1. If all pallets for node E are removed from route v_2, in the LS one also needs to remove arc (D, E) in route v_2.

If a customer is served by two different routes at the current iteration, a *1-0 exchange* move between these routes may lead to unsplit the service to this customer in the next iteration.

In Fig. 2 the caption (v_1, v_2, n_1, n_2) describes a *1-1 exchange* move, in which all pallets in vehicle v_2 for customer n_2 are swapped with all pallets in route v_1 for customer n_1. In the left side of the figure, the current solution is reported; in the right side, we show the solution after the local move. The following cases can occur:

case (a) Customer D in route v_2 is swapped with customer A in route v_1. Thus, in route v_2 one must remove arcs (C, D) and (D, E) and add arcs (C, A) and (A, F). In route v_1 one must remove arcs (s, A) and (A, B) and add arcs (s, D) and (D, B).

case (b) Customer E in route v_2 is swapped with customer B in route v_1. Thus, in route v_2 one must remove arc (D, E) and add arc (D, B). In route v_1 one must remove arc (A, B) and add arc (A, E).

case (c) Customer E in route v_2 is swapped with customer A in route v_1. Thus, in route v_2 one must remove arc (D, E) and add arc (D, A). In route v_1 one must remove arcs (s, A) and (A, B) and add arcs (s, E) and (E, B).

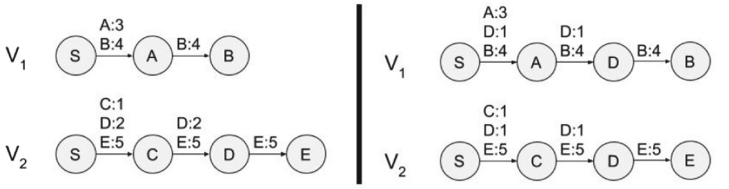
case (d) Customer D in route v_2 is swapped with customer B in route v_1. Thus, in route v_2 one must remove arcs (C, D) and (D, E), and add arcs (C, B) and (B, E). In route v_1 one must remove arc (A, B) and add arc (A, D).

If a customer is served by two different routes at the current iteration, a *1-1 exchange* move may result in the double occurrence of a customer in a route. Since it is clearly uneconomical to serve a customer twice in the same route, an additional move is performed to reach this customer only once in the route.

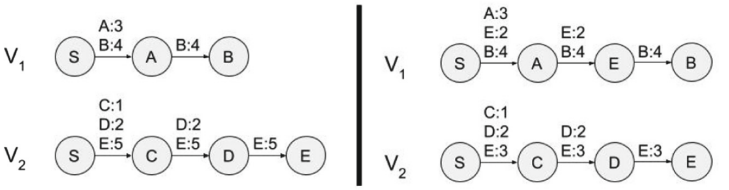
The following notation is introduced to present the TS algorithm:

- α is the number of initial solutions;
- S_j^0 is the jth initial solution;
- t_{run} is the running time of the TS algorithm;
- t_{max} is the maximum running time of the TS algorithm;
- S_j is the current solution built from S_j^0;
- $z(S_j)$ is the value of solution S_j;
- S_j^* is the best current solution derived from S_j^0;
- z_j^* is the value of solution S_j^* built from S_j^0;
- it is the number of iterations without any improvement;
- it_{max} is the maximum number of iterations without any improvement;
- $N(S_j)$ is the overall neighborhood of S_j (i.e. the set of solutions obtained both tabu and non-tabu moves to S_j);
- $\overline{N}(S_j)$ is the subset of $N(S_j)$ containing the solutions obtained by applying only non-tabu moves to S_j;

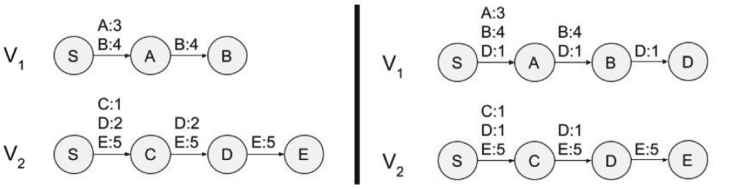
(a) *(v_1, v_2, A, D, 1)*



(b) *(v_1, v_2, A, E, 2)*



(c) *(v_1, v_2, B, D, 1)*



(d) *(v_1, v_2, B, E, 2)*

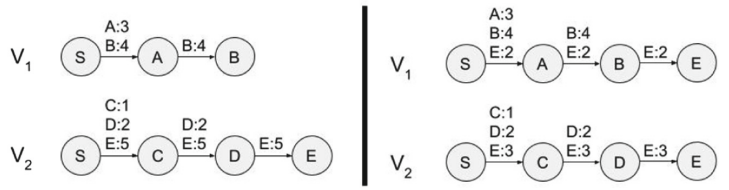
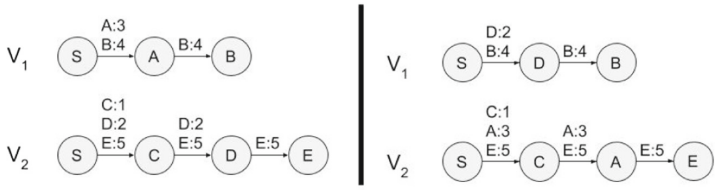
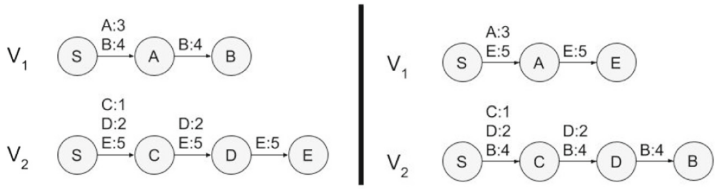


Fig. 1. Four possible (*1-0 exchange*) moves

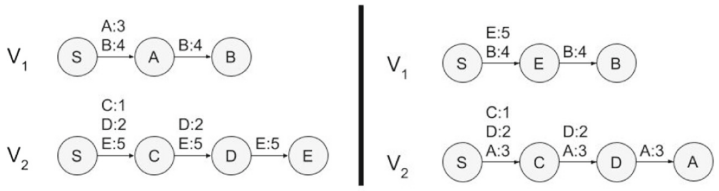
(a) *(v_1, v_2, A, D)*



(b) *(v_1, v_2, B, E)*



(c) *(v_1, v_2, A, E)*



(d) *(v_1, v_2, B, D)*

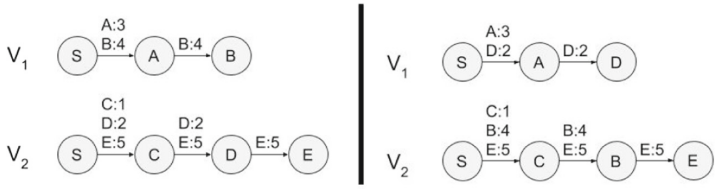


Fig. 2. Four possible (*1-1 exchange*) moves

– T: the Tabu List;

This is the template of the Tabu Search algorithm:

Construct α initial solutions $S_1^0 ... S_\alpha^0$; $j = 1$;
while $j \leq \alpha$ *and* $t_{run} \leq t_{max}$ **do**
{
  set $S_j \leftarrow S_j^0$, $z_j^* \leftarrow z(S_j^0)$, $S_j^* \leftarrow S_j^0$, $T \leftarrow \emptyset$, $it = 0$;
  while $it \leq it_{max}$ **do**
  {
    select S_j in $argmin_{S'_j \in \overline{N}(S_j)}[z(S'_j)]$;
    if $z(S_j) < z_j^*$ **then**
    {
      $z_j^* \leftarrow z(S_j)$; $S_j^* \leftarrow S_j$;
    }
    else
    {
      it=it+1;
    }
    record tabu for the current move in T;
    delete the oldest entries from T;
  }
  $j = j + 1$; update t_{run};
}
return S_k^* such that $z_k^* = min_{j=1,...,\alpha} z*_j$;

Algorithm 1: Tabu Search

The stopping criteria of the overall algorithm are the maximum running time t_{max} and the conclusion of the Tabu Search steps for all initial solutions. For a given initial solution, the non-tabu moves are made until the number of iterations without any improvement is lower or equal to it_{max}. The algorithm returns the best solution among those determined from all initial solutions.

Two parameters need to be calibrated according to the former scheme of the algorithm: α and it_{max}. Moreover, the tabu tenure β is set to be equal to it_{max}. Moreover, a parameter denoted by γ is adopted to control the size of the neighbourhood. When it is set to *AND*, the neighbourhood contains all solutions obtained by applying both *1-0 exchange* and *1-1 exchange* moves to the current solution. When it is set to *OR*, the neighbourhood contains all solutions obtained by applying a *1-0 exchange* move in an iteration, a *1-1 exchange* move in the next iteration and so on.

3.2 Tabu Search with Penalties

In this problem, the former definition of the search space may restrict the search too much and lead to low-quality solutions. More precisely, the route capacity may be too tight to allow effective LS search moves between two different routes. Therefore, we relax the capacity constraint and create a larger search space that

can be visited according to the moves described in the former section. Hence, the capacity constraint is dropped from the search space definition and weighted penalties for constraint violation are added to the objective. Clearly, this raises the issue of finding suitable weights for the constraint violation. The issue is faced by the calibration of two parameters:

- δ: it is the penalty (in percentage) to apply to the total cost of the infeasible solutions;
- ϵ: it is the percent increase in the capacity of vehicles.

4 Experimentation

The objectives of this section are twofold. The first objective is to show to what extent the problem can be solved by an off-the-shelf MILP solver. The second objective is to investigate the effectiveness of former variants of the Tabu Search for solving the problem. All tests have been implemented in Python and solved by the MILP solver Cplex 12.9 running with default parameters, except a maximum running time of 1 h and a required relative gap of 1e-4 (i.e. 0.01%). The experiments were performed on a 64 Gigabyte Ubuntu 16.04.6 LTS virtual machine running on a physical computer Cisco UCS B480 M5 Intel Xeon Gold 6136 3.00 GHz 12-core CPU equipped with 1 Terabyte of DDR4 RAM and 10 Terabytes SAS Hard drives.

The outcomes are reported in Table 1 and 2. All tables are divided into three groups of columns denoted by PROBLEM DATA, MILP and a variant of the Tabu Search algorithm: TS_B for the Basic Tabu Search in Table 1 and TS_P for the Tabu Search with penalties in Table 2. Since the problem is new, we generate several instances for the experimentation. The columns under PROBLEM DATA and MILP are identical in both tables. They report the name of the problem instance, the number of destinations ($|K|$), the number of vehicles ($|V|$), the overall number of pallets requested by all destinations ($\sum_{k \in K} d_k$) and the available transportation capacity ($\sum_{v \in V} u_v$). For example, instance P_1 has 6 customers, 2 vehicles, 70 pallets to be delivered and an overall transportation capacity of 80 pallets.

The group of columns denoted by MILP reports some outcomes on the solutions obtained by Cplex. More precisely, we indicate the time spent by Cplex to determine the best solution (t_{best}), the overall execution time (t_{exe}) and the relative optimality gap between the best solution and the lower bound (gap). For example, instance P_5 was solved to optimality after 2304, 2 seconds, it was solved to optimality, but the best solution was already determined after 244, 12 seconds. Therefore, the MILP solver determined this solution in a few minutes, but it took about 2000 additional seconds to prove its optimality. Cplex optimally solved all instances with less than 17 customers. In the range between 20 and 50 customers, one typically obtains a feasible solution without proving its optimality. Moreover, it is not possible to obtain a feasible solution for instances with more than 60 customers in one hour (these solutions are denoted by "UNSOLVED" in the tables). Therefore different solution methods must be adopted for the largest

instances of this problem. Hence, it is of interest to evaluate if they can be solved by the Tabu Search and evaluate its computational effectiveness.

The following configurations of the *Basic Tabu Search* were tested:

1. $\alpha = 20$, $\beta = 10$, $\gamma = OR$;
2. $\alpha = 50$, $\beta = 5$, $\gamma = OR$;
3. $\alpha = 50$, $\beta = 5$, $\gamma = AND$;
4. $\alpha = 10$, $\beta = 30$, $\gamma = OR$;
5. $\alpha = 20$, $\beta = 10$, $\gamma = AND$.

Each instance was run in the server according to each configuration for one hour (i.e. t_{max} was also set to 1 hour). In Table 1 we denote by gap_{avg} the average relative gap between these configurations and the best solution determined by the MILP solver. Moreover, gap_{best} is the best gap among all configurations and $cnfg_{best}$ is the index of the best configuration according to the former list. For example, in problem P_2, the best configuration is the third in the former list.

Table 1. Computational experiments on the Basic Tabu Search

PROBLEM DATA					MILP			TS_B		
Instance	$\|K\|$	$\|V\|$	$\sum_{k\in K} d_k$	$\sum_{v\in V} u_v$	t_{best}	t_{exe}	gap_{opt}	gap_{avg}	gap_{best}	$cnfg_{best}$
P_1	006	2	70	80	0,18	0,04	0,00	0,00	0,00	all
P_2	009	4	78	80	34,81	34,81	0,00	2,08	0,99	3
P_3	010	8	113	120	87,40	820,13	0,00	3,58	2,65	2
P_4	015	2	117	120	150,42	212,15	0,00	0,18	0,00	3
P_5	016	3	176	195	244,12	2304,2	0,00	4,96	4,29	2
P_6	017	2	181	200	83,5	185,56	0,00	3,28	3,03	2
P_7	019	3	212	250	630	3600	1,81	1,47	0,82	3
P_8	020	3	202	210	1756,58	3600	6,10	5,34	1,96	3
P_9	030	4	389	400	520,69	1057,23	0,00	41,20	28,55	2
P_{10}	040	4	370	375	2100	3600	19,21	38,41	29,79	2
P_{11}	050	3	533	550	2400	3600	22,69	33,04	12,76	2
P_{12}	060	3	649	675	UNSOLVED			SOLVED		2
P_{13}	070	4	806	810	UNSOLVED			SOLVED		2
P_{14}	080	2	804	850	UNSOLVED			SOLVED		2
P_{15}	090	4	914	925	UNSOLVED			SOLVED		2
P_{16}	100	6	1030	1050	UNSOLVED			SOLVED		2

Generally speaking, the *Basic Tabu Search* can solve all instances within one hour. As for the largest instances, we cannot compute any gap, as no comparison is possible with respect to the MILP solver. Nevertheless, we denote these solutions by the string "SOLVED", to point out the ability to determinate feasible solutions. The best calibrations are those denoted by 2 and 3. Therefore, the number of initial solutions is the most beneficial parameter for the quality of final solutions.

The outcomes exhibit near optimal gaps w.r.t the MILP solver in the case of small instances. Clearly, some gaps become clearly high in the case of medium instances (e.g. in instances P_9, P_{10} and P_{11}) and they need to be decreased. In what follows, we investigate if some improvement is obtained by the *Tabu Search with Penalties*. The following configurations of TS_P were investigated:

a. $\delta = 25\%$, $\epsilon = 25\%$;
b. $\delta = 10\%$, $\epsilon = 25\%$;
c. $\delta = 25\%$, $\epsilon = 10\%$;
d. $\delta = 10\%$, $\epsilon = 10\%$;
e. $\delta = 5\%$, $\epsilon = 10\%$;
f. $\delta = 5\%$, $\epsilon = 5\%$.

These configurations were combined with those on TS_B and result in 30 calibrations. They are denoted by a pair of indices, each taken from the list of configurations. For example, configuration $2f$ means $\alpha = 50$, $\beta = 5$, $\gamma = OR$, $\delta = 5\%$, $\epsilon = 5\%$. The best calibration is reported in column $cnfg_{best}$.

Table 2. Computational experiments on the Tabu Search with penalties

PROBLEM DATA					MILP			TS_P		
Instance	$\|K\|$	$\|V\|$	$\sum_{k\in K} d_k$	$\sum_{v\in V} u_v$	t_{best}	t_{exe}	gap_{opt}	gap_{avg}	gap_{best}	$cnfg_{best}$
P_1	006	2	70	80	0,18	0,04	0,00	0,00	0,00	all
P_2	009	4	78	80	34,81	34,81	0,00	4,47	0,97	2f
P_3	010	8	113	120	87,40	820,13	0,00	7,58	1,08	5d
P_4	015	2	117	120	150,42	212,15	0,00	1,31	0,00	1d, 2e, 3c, 3d, 3e, 5a
P_5	016	3	176	195	244,12	2304,2	0,00	9,22	3,50	1f
P_6	017	2	181	200	83,5	185,56	0,00	6,36	0,93	2b
P_7	019	3	212	250	630	3600	1,81	6,65	0,80	2a, 4b
P_8	020	3	202	210	1756,58	3600	6,10	4,63	0,00	1e, 2e, 4c
P_9	030	4	389	400	520,69	1057,23	0,00	57,05	25,98	3f
P_{10}	040	4	370	375	2100	3600	19,21	49,30	29,74	2f
P_{11}	050	3	533	550	2400	3600	22,69	34,12	3,75	3f
P_{12}	060	3	649	675	UNSOLVED			SOLVED		5f
P_{13}	070	4	806	810	UNSOLVED			SOLVED		4f
P_{14}	080	2	804	850	UNSOLVED			SOLVED		1f
P_{15}	090	4	914	925	UNSOLVED			SOLVED		3f
P_{16}	100	6	1030	1050	UNSOLVED			SOLVED		2f

The outcomes show that TS_P can improve the gaps w.r.t TS_B, even if some room for improvement does exist in the solutions of medium sized instances. Moreover, the best outcomes are obtained in the case $\delta = 5\%$ and $\epsilon = 5\%$, which means that limited violated capacities and low penalization costs are typically recommended. In both tables, the execution of the Tabu Search algorithm is stopped by the number α of initial solutions in instances $P_1, ..., P_8$ (i.e. a local optimum is joined for $\beta = it_{max}$ times for each initial solution). The execution of the other instances is stopped by t_{max}.

5 Conclusions

The efficient management of the freight flows is one of the key pillars for sustainable growth of cities. Urban mobility managers must adopt smart transport solutions within the framework of the City Logistics to meet the biggest challenges that cities are facing today (e.g. traffic and congestion). Generally speaking, ICT tools must be developed to overcome socio-economic problems and must be coupled with Operations Research methods to optimize the operations.

In this paper we investigated a problem motivated by the missing implementation of ICT tools, that can share knowledge on disaggregated loads in containers (or semi-trailers) entering a port city. This knowledge can be exploited to determine the intermediate facilities (or satellites) where containers can be cross-docked, in order to start the last-mile distribution of these loads toward their final destinations.

We aimed to determine the paths of vehicles from a selected satellite to the destinations, as well as the paths of the loads carried by these vehicles. Since the transportation requests are highly customized, the problem was described as a service network design problem, in which each request is a commodity. This characteristic allows to evaluate the tradesoffs between the costs of vehicles and those generated by the volumes of the different loads moved by vehicles. The added value of this paper is the explicit consideration of the last costs, which are typically ignored by planning tools based on Vehicle Routing Problems. The information on the flows of each commodity can be adopted by urban mobility managers to support the decision processes on planning or improving facilities, the definition of new transportation services and the allocation of (human and material) resources to tasks [16].

The objective of this paper was the minimization the overall costs faced by carriers, customers and cities, while satisfying all the transportation requests by the available vehicles. The problem was faced by Operations Research methods. First, a Mixed Integer Linear Programming Model was formulated for this problem. Next, a Tabu Search metaheuristic was presented as a solution method to solve larger problem instances.

The research in the field is in progress. The concept of granularity will be investigated to reduce the size of the neighborhood and, thus, increase the number of initial solutions in the Tabu Search algorithm. Moreover, we will present a full city-logistics model, which will also incorporate decisions on the selection of vehicles and satellites. Finally, it will be worth comparing the outcomes of this paper with respect to the current *modus operandi* of mobility managers.

References

1. United Nations: World urbanization prospects, the 2014 revision. Technical report, United Nations, Department of Economic and Social Affairs, Population Division, New York, USA (2014)

2. Bektas, T., Crainic, T.G., Van Woensel, T.: From managing urban freight to smart city logistics networks. In: Gakis, K., Pardalos, P. (eds.) Networks Design and Optimization for Smart Cities (Series on Computers and Operations Research), vol. 8, pp. 143–188. World Scientific Publishing, Singapore (2017)
3. Savelsbergh, M., Van Woensel, T.: City logistics: challenges and opportunities. Transp. Sci. **50**(2), 579–590 (2016)
4. Crainic, T.G., Ricciardi, N., Storchi, G.: Models for evaluating and planning city logistics systems. Transp. Sci. **43**(4), 432–454 (2009)
5. Paddeu, D.: Sustainable solutions for urban freight transport and logistics: an analysis of urban consolidation centers. In: Zeimpekis, V., Aktas, E., Bourlakis, M., Minis, I. (eds.) Sustainable Freight Transport. ORSIS, vol. 63, pp. 121–137. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-62917-9_8
6. Ignaccolo, M., Inturri, G., Giuffrida, N., Torrisi, V., Cocuzza, E.: Sustainability of freight transport through an integrated approach: the case of the eastern sicily port system. Transp. Res. Proc. **45**, 177–184 (2020)
7. Crainic, T.G., Kim, K.H.: Intermodal transportation. In: Henderson, S.G., Nelson, B.L. (eds.) Handbooks in Operations Research and Management Science, vol. 14, pp. 467–537. Elsevier, Amsterdam (2007)
8. Calabrò, G., Torrisi, V., Inturri, G., Ignaccolo, M.: Improving inbound logistic planning for large-scale real-world routing problems: a novel ant-colony simulation-based optimization. Eur. Transp. Res. Rev. **12**, 1–11 (2020). https://doi.org/10.1186/s12544-020-00409-7
9. Wen, M., Larsen, J., Clausen, J., Cordeau, J.F., Laporte, G.: Vehicle routing with cross-docking. J. Oper. Res. Soc. **60**(12), 1708–1718 (2009). https://doi.org/10.1057/jors.2008.108
10. Roso, V., Woxenius, J., Lumsden, K.: The dry port concept: connecting container seaports with the hinterland. J. Transp. Geogr. **17**(5), 338–345 (2009)
11. Crainic, T.G., Dell'Olmo, P., Ricciardi, N., Sgalambro, A.: Modeling dry-port-based freight distribution planning. Transp. Res. Part C Emerg. Technol. **55**, 518–534 (2015)
12. Tavakkoli-Moghaddam, R., Safaei, N., Kah, M.M.O., Rabbani, M.: A new capacitated vehicle routing problem with split service for minimizing fleet cost by simulated annealing. J. Franklin Inst. **344**(5), 406–425 (2007)
13. Archetti, C., Campbell, A.M., Speranza, M.G.: Multicommodity vs. single-commodity routing. Transp. Sci. **50**(2), 461–472 (2016)
14. Cattaruzza, D., Absi, N., Feillet, D., Vigo, D.: An iterated local search for the multi-commodity multi-trip vehicle routing problem with time windows. Comput. Oper. Res. **51**, 257–267 (2014)
15. Gendreau, M.: An introduction to tabu search. In: Glover, F., Laguna, M. (eds.) Handbbok of Metaheuristics, pp. 37–54. Springer, Boston (2003). https://doi.org/10.1007/0-306-48056-5_2
16. Crainic, T.G.: Service network design in freight transportation. Eur. J. Oper. Res. **122**(2), 272–288 (2000)