A Graph Reduction Heuristic For Supply Chain Transportation Plan Optimization

Simon Belieres, Nicolas Jozefowiez, Frédéric Semet

A Graph Reduction Heuristic For Supply Chain Transportation Plan Optimization

Simon Belieres
LAAS-CNRS
Université de Toulouse, Toulouse, France
Email: sbeliere@laas.fr

Nicolas Jozefowiez
LCOMS
Université de Lorraine, Metz, France

Frédéric Semet
Centre de Recherche en Informatique, Signal et Automatique, Lille
Ecole Centrale de Lille, Lille, France

1 Introduction

This paper describes an algorithm for the computation of tri-echelon supply chains transportation plans. The problem is inspired from logistic issues appearing in large-scale restaurant chains. We consider products for which customers have a recurring request through the time horizon. Suppliers product and ship the commodities to customers by direct deliveries, or by way of intermediary hubs.

Freight flows in the supply chain are modelled by time-expanded graphs [3]. Unfortunately, realistic instances in terms of network and time horizon yield to graphs with a huge amount of nodes and arcs. Induced mixed integer linear programs are too large to be solved in a reasonable amount of time with an industrial solver.

We conceived a resolution heuristic able to produce good solutions despite instances increasing size. The method is inspired from Boland et al. [1]. The authors focus on the Service Network Design Problem ($SNDP$). It is a comparable freight transportation problem, reviewed by Crainic [2]. However, the method cannot be applied directly as our problem differs on key elements. Our heuristic consists in generating a subgraph containing nodes and arcs required to transit the optimal flow. When generation is over, we solve a much smaller program induced from the subgraph, and get a not neccesarily optimal transportation plan.

2 Problem description

The static network $\mathcal{D} = (\mathcal{N}, \mathcal{A})$ models our supply chain. The nodes $\mathcal{N}$ represents the actors of the supply chain, partitioned in three sets. The set $\mathcal{U}$ represents the suppliers, $\mathcal{V}$ the hubs, and $\mathcal{W}$ the restaurants. We consider a set of commodities $\mathcal{K}$. Each supplier $u \in \mathcal{U}$ can infinitly provide a subset of products $\mathcal{K}^u \in \mathcal{K}$. Each restaurant $w \in \mathcal{W}$ has a demand $d^t_{wk} \geq 0$ of commodity k at time t. Our objective is to determine the minimal cost transportation plan satisfying all the demands.

There is no temporal dimension in static networks, so we represent the supply chain by a time-expanded graph $\mathcal{D}_T = (\mathcal{N}_T, \mathcal{H}_T \cup \mathcal{A}_T)$ derived from $\mathcal{D}$. The set of nodes $\mathcal{N}_T$ is obtained duplicating the physical locations of $\mathcal{N}$ through the time horizon. The set of arcs is decomposed into holding arcs $\mathcal{H}_T$ - connecting two occurences of the same physical location - and transportation arcs $\mathcal{A}_T$. Note that holding arcs only exist for hubs. Each arc $((i,t),(j,t'))$ has a travel time $t'-t$, a per-unit-of-flow cost $c_{ij} \in \mathbb{R}^{+*}$, a fixed cost $f_{ij} \in \mathbb{R}^{+*}$, and a capacity $u_{ij} \in \mathbb{R}$.

For each demand d^t_{wk} the destination is known, but the origin is unknown. Indeed, any supplier $u \in \mathcal{U}$ such that $k \in \mathcal{K}^u$ can fulfill the request. The solver must determine which one is the best option. Thus we have no commodities origin constraints, involving an unusual complexity level for transportation problems.

Given a time-expanded network $\mathcal{D}_T$, we define $SCNDP(\mathcal{D}_T)$ to be our Supply Chain Network Design Problem. Positive integer variables $y^{tt'}_{ij}$ represent the number of trucks used on arc $((i,t),(j,t'))$. Positive continuous variables $x^{ktt'}_{ij}$ model the flow of commodity k on arc $((i,t),(j,t'))$. Note that $x^{ktt'}_{ij}$ is not defined if k cannot transit on the given arc, i.e. if $i \in \mathcal{U}$ and $k \notin \mathcal{K}^i$. The following is a valid integer programm:

$$\min\ z(\mathcal{D}_T) = \sum_{\mathcal{A}_T} f_{ij} y^{tt'}_{ij} \quad + \sum_{k \in \mathcal{K}} \sum_{\mathcal{A}_T} c_{ij} x^{ktt'}_{ij} \quad + \sum_{k \in K} \sum_{\mathcal{H}_T} c_{ii} x^{ktt'}_{ii}$$

$$\sum_{\mathcal{A}_T \cup \mathcal{H}_T} x^{ktt'}_{ij} - \sum_{\mathcal{A}_T \cup \mathcal{H}_T} x^{ktt't''}_{jl} \begin{cases} = 0 & \forall (j,t) \in \mathcal{V}_T \\ \geq d^t_{wk} & \forall (j,t) \in \mathcal{W}_T \end{cases} \quad \forall k \in \mathcal{K},\ \forall (i,t) \in \mathcal{N}_T \tag{1}$$

$$\sum_{k \in \mathcal{K}} x^{ktt'}_{ij} \leq u_{ij} y^{tt'}_{ij} \quad \forall ((i,t),(j,t')) \in \mathcal{A}_T \tag{2}$$

$$x^{ktt'}_{ij} \in \mathbb{R}^{+} \quad \forall ((i,t),(j,t')) \in \mathcal{A}_T \cup \mathcal{H}_T,\ \forall k \in \mathcal{K} \tag{3}$$

$$y^{tt'}_{ij} \in \mathbb{N}^{+} \quad \forall ((i,t),(j,t')) \in \mathcal{A}_T \tag{4}$$

We seek to minimize the total expenses, i.e. the fixed costs of allocating resources on transportation arcs and the linear costs of transportation and holding flows. Constraint (1) is an adapted Kirchhoff constraint, with no imposed origin on commodities. Constraint (2) ensures that enough trucks are allocated to ship the commodities. Constraints (3) and (4) are the variable domains.

3 Resolution method

3.1 Dynamic Discrezation Discovery

Dynamic Discretization Discovery (DDD), method by Boland et al., is optimal for the Service Network Design Problem (SNDP). This problem is comparable to ours, with two major differences: commodities origins are defined and holding cost are null. The objective function seeks to minimize transportation costs only.

The DDD method consists in generating an initial subgraph D_T under a certain set of properties. Those conditions respected, the subproblem $SNDP(D_T)$ provides a solution not necessarily feasible in the full graph. The subgraph contains arcs under-estimating real travel times, which allow commodities to arrive earlier than feasible in real-life. Therefore the subgraph offers unrealistic flow consolidations. However, the subproblem necessarily provides a lower bound to the initial SNDP.

The DDD method solves the subproblem $SNDP(D_T)$ and detects the use of early-arrival arcs in the solution. In that case, the subgraph is repaired and expanded while keeping its lower bound propery. The process stops once a lower bound feasible to the full graph is found, i.e. the optimal solution.

3.2 A heuristic based on two reparation mechanisms

The DDD cannot be used unaltered to our problem, essentially because we consider strictly positive holding costs and our commodities have no predefined origins.

Considering strictly positive holding costs breaks the optimality property of the DDD, as we set an example in which the subproblem no longer provide a lower bound to the full problem. This is because having unrealistic flow consolidations is not sufficient to ensure finding a better solution than possible. Supressing the commodities origins prevent us from generating the initial subgraph similarly to the DDD.

We demonstrate the DDD is optimal to the SCNDP with free holding costs, if the subgraph contain any supplier occurence (u, t). However we must restrict the set of suppliers size, as including them all makes the method too slow. We initiate the subgraph with the suppliers shipping a non-null amount of commodities in the optimal solution of the SCNDP linear relaxation.

The subgraph generation heuristic is based on two reparation mechanisms. We build the initial subgraph, with early-arrival arcs and free holding costs only. The first reparation mechanism is the DDD, it fixes the transportation unfeasibilities. It detects the early-arrival shippings and refine the subgraph to prevent unfeasible consolidations. The second reparation mechanism is an holding costs injection, to fix the storage defects. It spots the free holding arcs used in the subgraph solution, and update the real costs. The method iterates those mechanisms until no transportation/storage anomaly appears. Note that

the programs solved in this phasis are linear relaxations only, to accelerate the process.

Then, the subgraph remaining early-arrival arcs are supressed and all the holding costs are updated. We then solve the SCNDP mixed-integer program induced from the subgraph, and obtain a feasible solution.

4 Results and Discussion

We tested the heuristic on 2 set of instances - representing a total of 90 instances - and compared it with the solution of the full SCNDP model by Gurobi. The first set is referred as easy instances and the industrial solver found optimal solutions, within the time limit of 2 hours. The second set is referred as difficult instances and the industrial solver only gave suboptimal solutions. The following ratio compares the results:

$$\text{ratio} = \frac{\text{H}_{sol} - \text{MILP}_{sol}}{\text{MILP}_{sol}} \times 100$$

A negative ratio states the heuristic outperformed Gurobi, otherwise Gurobi found a better solution.

$\mathcal{N}$	Ratio(%)
10	0.70
20	2.19
30	1.13

$\mathcal{N}$	Ratio(%)
80	-0.056
100	-18.75
120	-28.17

Table 1: Performance on easy/difficult instances

To summarize, we agglomerated the instances by the number of nodes of the supply chain. For easy instances, the average ratio is positive as Gurobi finds optimal solutions, what the heuristic does not. However, the ratio values are close to 0, indicating the heuristic solutions are of good quality. The difficult instances table reveals our heuristic is able to provide better solution than Gurobi in a given computational time. We observe that this gap becomes larger as the instance complexity increases.

References

[1] Boland N., Hewitt M., Marshall L., Savelsbergh M. (2017). *The Continuous Time Service Network Design Problem.* Operations Research, 65(5), 1303-1321.

[2] Crainic T. G. (2000). *Service Network Design in freight transportation.* European Journal of Operational Research, 122(2), 272-288.

[3] Ford Jr L. R., Fulkerson D. R. (1962). *Flows in networks.* Princeton university press.