20th International Congress on Modelling and Simulation, Adelaide, Australia, 1–6 December 2013
www.mssanz.org.au/modsim2013

Time Aggregation for Network Design to Meet Time-Constrained Demand

N. Boland[a], A. Ernst[b], T. Kalinowski[a], M. Rocha de Paula[a], M. Savelsbergh[a], and G. Singh[b]

[a]School of Mathematical and Physical Sciences, The University of Newcastle
[b]CSIRO Mathematics, Informatics and Statistics
Email: Martin.Savelsbergh@newcastle.edu.au

Abstract:

We study a network design problem inspired by a strategic planning problem encountered in the Hunter Valley Coal Chain. Demand is given in the form of freight that is available from a specific date and has to be transported from multiple origins to a single destination before its deadline. It is possible to temporarily store freight at certain intermediate locations along the way from origins to destination. The objective is to determine minimum-cost capacity expansions required on the links and nodes of the network, if any, so as to be able to transport all freight within its given time windows.

A natural mixed integer programming formulation with a daily granularity quickly becomes computationally intractable. We investigate the potential of time aggregation to overcome the computational challenges. By aggregating consecutive time periods, a smaller instance is obtained, which can be solved more easily and provides a lower bound on the optimal value.

A carefully designed iterative disaggregation scheme identifies a time aggregation that yields an optimal solution to the original problem. An extensive computational study demonstrates the efficacy of the proposed approach.

Keywords: *Network design problem, integer programming, time indexed models, time aggregation*

1 INTRODUCTION

In this research, we study a general class of network design problems that exhibit some of the essential features of the strategic planning problem encountered at the Hunter Valley Coal Chain (HVCC).

Consider a network with a set of nodes N and a set of arcs A and a set of commodities C that have to be transported over the network. The freight of commodity $j \in C$ can be temporarily stored in certain nodes $N_j^S \subseteq N$. Each commodity has an associated demand F_j that has to flow from a set of sources $Q_j \in N_j^S$ to a single destination $s_j \in N_j^S$. Each of the sources $q \in Q_j$ has an associated contribution f_j^q to the total demand. The freight for commodity $j \in C$ is available at its sources at release date r_j and must reach its destination at or before a deadline d_j. Each arc $a \in A$ has a current capacity u_a that can be expanded to a maximum of u_a^+ for a fixed cost of c_a^1 plus c_a^2 per unit of installed capacity. Similarly, every storage node $v \in (N^S = \bigcup_{j \in C} N_j^S)$ has also a current capacity u_v that can be expanded to a maximum of u_v^+ for a fixed cost of c_v^1 plus c_v^2 per unit of installed capacity. The objective is to determine minimum-cost capacity expansions required on the links and nodes of the network, if any, so as to be able to transport all freight within its given time windows.

Due to their great importance to industry, Capacity Expansion Problems (CEPs) have been studied since the 60's, with many different applications, as shown in an early survey published by Luss (1982). Ordóñez and Zhao (2007) consider the expansion of arc capacities in a network to minimize the linear transportation cost in a network with uncertain data. Ahmed et al. (2003) consider a similar problem, exploiting the lot sizing substructure of their problem, also in an uncertain environment, to minimize the cost of cost of the expansions.

In section 2, we formulate the CEP using a natural Mixed-Integer Programming (MIP) model considering discrete time periods. As discussed by Floudas and Lin (2004), when using this kind of time representation, it is usually necessary to use time periods that are sufficiently small to achieve a suitable approximation of the real world problem. That, along with the use of long time horizons, leads to very large optimization problems which are often intractable.

To cope with that, Newman and Kuchta (2007) collapsed adjacent time periods into stages with the same duration (the same number of time periods) to reduce the size of a model used to schedule iron ore production over multiple discrete time periods. In this research, we generalize this idea, proposing the use of smaller problems that aggregate an arbitrary number of time periods, to obtain lower bounds for the problem.

In section 3, we propose a carefully designed iterative disaggregation scheme to identify a time aggregation that yields an optimal solution to the original problem.

Finally, in section 4 we present computational results obtained with the proposed disaggregation schemes on instances derived from real HVCC data, and conclude with some final remarks in section 5.

2 A NATURAL MIP FORMULATION WITH DISCRETE AGGREGATED TIME PERIODS

In this research, the proposed models consider that the time horizon $T = [min_{j \in C}(r_j), max_{j \in C}(d_j)]$ is discretized in $(max_{j \in C}(d_j) - min_{j \in C}(r_j) + 1)$ time periods with the same duration.

Define a time aggregation a set of adjacent time periods. Let a breakpoint be the first time period of a time aggregation and define $\beta \subseteq T$ as the ordered list of such breakpoints. Let $\bar{t} \in \beta = \{..., t, \bar{t}, ...\}$ be the breakpoint found immediately after breakpoint t in β. A time aggregation is thus defined by two consecutive breakpoints $[t, \bar{t}) \in \beta$.

Formulation 1 formulates the CEP accounting for time aggregations.

In this model, the binary design decision variables y_a and y_v are set to 1 if we have to install extra capacity on arc $a \in A$ or storage node $v \in N^S$, respectively, and 0 otherwise. The design decision variables z_a and z_v hold the extra capacity installed on arc a or storage node v, respectively. The operational decision variables ϕ_{aj}^t and ϕ_{vj}^t hold the flow of commodity $j \in C$ at time $t \in [r_j, d_j]$ on arc a or storage node v, respectively.

Constraints (2) and (3) ensure that capacity is installed only on selected items, and for these there is an upper bound for expansion. Constraints (4) and (5) are flow conservation constraints. They also ensure that each commodity only flows during its respective time window. Constraints (6) ensure that the correct amounts of flow are sent from the sources, starting at the correct time. Constraints (7) and (8) are capacity constraints.

Aggregated problems can be of interest in practice because of their reduced size, which would allow them to be solved quicker than the original problem, and because they can provide us with interesting information about the instance or even solve the original problem.

$$\min \sum_{a\in A}(c_a^1 y_a + c_a^2 z_a) + \sum_{v\in N^S}(c_v^1 y_v + c_v^2 z_v) \tag{1}$$

s.t.

$$z_a \leq u_a^+ y_a \qquad \forall a \in A \tag{2}$$

$$z_v \leq u_v^+ y_v \qquad \forall v \in N^S \tag{3}$$

$$\sum_{a\in\delta^-(v)} \phi_{aj}^t - \sum_{a\in\delta^+(v)} \phi_{aj}^t = 0 \qquad \forall j \in C, v \in N\backslash N^S, t \in [r_j, d_j] \cap \beta \tag{4}$$

$$\sum_{a\in\delta^-(v)} \phi_{aj}^t - \sum_{a\in\delta^+(v)} \phi_{aj}^t = \phi_{vj}^t - \phi_{vj}^{t-1} \qquad \forall j \in C, v \in N\backslash N_j^S, t \in [r_j, d_j] \cap \beta \tag{5}$$

$$\sum_{a\in\delta^+(q)} \phi_{aj}^t + \phi_{qj}^t = f_j^q \qquad \forall j \in C, q \in Q_j, t \in \beta : r_j \in [t, \bar{t}) \tag{6}$$

$$\sum_{j\in C: t\in[r_j,d_j]} \phi_{aj}^t \leq (\bar{t} - t)(u_a + z_a) \qquad \forall a \in A, t \in \beta \tag{7}$$

$$\sum_{j\in C: t\in[r_j,d_j]} \phi_{vj}^t \leq (\bar{t} - t)(u_v + z_v) \qquad \forall v \in N^S, t \in \beta \tag{8}$$

$$y_a, y_v \in \{0, 1\} \tag{9}$$

$$z_a, z_v \geq 0 \tag{10}$$

$$\phi_{aj}^t \geq 0 \tag{11}$$

Formulation 1. A natural time-expanded formulation for the Network Design Problem with $|T|$ time periods, accounting for time aggregations. Define $\beta \subseteq T$ as the ordered list of such breakpoints and let $\bar{t} \in \beta = \{..., t, \bar{t}, ...\}$ be the breakpoint found immediately after a breakpoint t in β.

Theorem 1. *The value of an optimal solution to an aggregated problem provides a lower bound on the value of an optimal solution to the fully disaggregated problem.*

Proof. Consider a set of breakpoints $\beta = \{...t, \bar{t}, ...\}$. From (6), if $r_j > t \in [t, \bar{t})$, then j is allowed to be dispatched at t rather than r_j. In the same way, if $d_j < (\bar{t} - 1) \in [t, \bar{t})$, then j is allowed to be delivered at $\bar{t} - 1$ rather than d_j. Since in this case $(\bar{t} - t) \geq (r_j - d_j)$, it follows that $(\bar{t} - t)\sum_{a\in A}(u_a + z_a) + (\bar{t} - t)\sum_{v\in N^S}(u_v + z_v) \geq (r_j - d_j)\sum_{a\in A}(u_a + z_a) + (r_j - d_j)\sum_{v\in N^S}(u_v + z_v)$. Therefore, since there could already be "extra" capacity being considered in the aggregated problem, we might not have to expand it as much as we would have to in the original problem. That would thus incur a smaller cost (objective function) for the obtained solutions. □

2.1 COMMODITY SHORTFALLS

After solving an aggregated problem, we obtain tentative values for the design variables (y and z). These values are tentative because the installed capacity might not be sufficient to accommodate all the demand in the most congested time periods. To check if the tentative solution is feasible for the original problem, we can maximize the flow in the original network (with all time periods) for the tentative design, by fixing the design variables obtained with the aggregation and setting the objective function to maximize the sum of the flows going out of all source nodes in all time periods.

Equation 12 gives the portion of the demand of commodity $j \in C$ that could not be delivered given the installed capacity in the network, i.e., the shortfall

$$\delta_j = F_j - \Big(\sum_{a\in\delta^-(s_j)} \tilde{\phi}_{aj}^{d_j} + \tilde{\phi}_{s_j j}^{d_j - 1}\Big) \tag{12}$$

Theorem 2. *If $\delta_j = 0\ \forall j \in C$, then the capacity expansions are optimal for the fully disaggregated problem.*

Proof. Because, unlike the aggregated model, the max flow model considers all time indices, and enforce the original time windows for all commodities, the total available capacity in this problem is the same as the total available capacity in the original problem. From (1), it follows that each source dispatches at most F_j for each commodity. If $\delta_j = 0 \forall j \in C$, it means that exactly F_j reached the target node at d_j for all commodities and, therefore, the installed capacities were enough to transport all demand in time. In other words, there exists ϕ such that the fully disaggregated model is feasible for this particular set of design variables y and z. Because of Theorem 1, in this case, the solution of the aggregation is feasible for the original problem, such a solution must be also optimal. □

3 A DISAGGREGATION SCHEME

Algorithm 1 proposes a disaggregation scheme that, at each iteration, analyzes the commodity shortfalls of an aggregation and refines the aggregation by introducing a new breakpoint (or a set of new breakpoints).

Consider the time criticality χ_t of a time period $t \in T$:

$$\chi_t = \sum_{j:t\in[r_j,d_j]} \delta_j \tag{13}$$

Note that whereas the commodity shortfall δ_j focuses on commodities, the time criticality χ_t focuses on time periods.

Because the value of χ can only change in time periods corresponding to a release date or a deadline of a commodity, breakpoints are only introduced at release dates and deadlines.

The algorithm stops either when the design variable values are feasible for the original problem ($\delta_j = 0 \; \forall j \in C$), as the solution will be optimal, or when a sufficient amount of freight can already flow with the incumbent installed capacity (where sufficient can be defined based on the specific setting and goals).

Algorithm 1: Constructive algorithm based on disaggregations. Let β be the set of selected breakpoints, and define $\overline{\beta} = \{t \in \{0..T\} : t = d_j \text{ or } t = r_j \text{ for some } j \in C\} \backslash \beta$. SolveAggregation($\beta$) creates and solves an aggregated problem considering the breakpoints in β, and yields optimal values $\tilde{y}$ and $\tilde{z}$ for the design variables. SolveMaxFlow($\tilde{y}, \tilde{z}$) creates and solves a max flow problem fixing the design variables y and z, and yields optimal flow values $\tilde{\phi}$ for its operation variables. FindBestCandidateBreakpoint($\overline{\beta}$) determines a candidate breakpoint for disaggregation.

$\beta = \{0\}$;
repeat
  $(\tilde{y}, \tilde{z}) \leftarrow$ SolveAggregation(β);
  $\tilde{\phi} \leftarrow$ SolveMaxFlow($\tilde{y}, \tilde{z}$);
  $\delta_j \leftarrow F_j - (\sum_{a\in\delta^-(s_j)} \tilde{\phi}^{d_j}_{aj} + \tilde{\phi}^{d_j-1}_{s_j j})$;
  if $\delta_j = 0 \; \forall \; j$ *or* $\sum_{j\in C}(F_j - \delta_j)/\sum_{j\in C} F_j > 0.95$ **then return** $(\tilde{y}, \tilde{z})$;
  $t \leftarrow$ FindBestCandidateBreakpoint($\overline{\beta}$);
  $\beta = \beta \cup \{t\}$;
until *Forever*;

3.1 FindBestCandidateBreakpoint($\overline{\beta}$)

There are many possible strategies to select a time period to disaggregate at. In this research, we explore analyzing the incumbent solution's commodity shortfall and time criticality.

The first approach attempts to find a problematic commodity to be "isolated" at each iteration, and disaggregating at their release and due date, hoping to (indirectly) improve the maximum possible flow. One way to characterize such a commodity is by calculating $argmax_{j\in C:\{r_j,d_j\}\cap\overline{\beta}\neq\emptyset}(\delta_j)$.

The second approach attempts to find a problematic time period directly and disaggregate around it. One way to characterize such a time period is by calculating $argmax_{t\in\overline{\beta}}\{\chi_t\}$.

4 COMPUTATIONAL EXPERIENCE

We use a test set with 18 instances divided in 6 sets with 3 different instances each, as detailed on Table 1. The instances are derived from real scenarios observed in the HVCC. The underlying network structure contains 104 nodes and 129 arcs. Every commodity can be stored in four of the nodes (not necessarily the same nodes) and the difference between release date and deadline is no more than five days.

Instance	Tonnage (Mt)	T	$\|C\|$	Instance	Tonnage (Mt)	T	$\|C\|$
1	90	120	336	10	90	360	990
2	90	120	327	11	90	360	994
3	90	120	331	12	90	360	997
4	140	120	503	13	140	360	1491
5	140	120	505	14	140	360	1474
6	140	120	512	15	140	360	1472
7	160	120	595	16	160	360	1666
8	160	120	568	17	160	360	1650
9	160	120	564	18	160	360	1662

Table 1. Description of the test set. Each row represents a set. The columns represent the set number, the total tonnage, the number of days in planning period, and the number of commodities. Each set comprises three different instances.

The proposed methods were coded in python and ran on an Intel Core i7 laptop operating at 3.1GHz with 8GB of RAM. All MIPs and LPs were solved using CPLEX (v12.5, using only the default settings).

Table 2 summarizes the results obtained when simply trying to solve the instances with CPLEX without any aggregation with a time limit of one hour.

Instance	Time (s)	Instance	Time (s)
1	8	10	181
2	7	11	169
3	7	12	139
4	54	13	*Timeout*
5	52	14	708
6	60	15	846
7	749	16	*Timeout*
8	2092	17	*Timeout*
9	1029	18	*Timeout*

Table 2. Results obtained by simply attempting to solve the instances under consideration without any aggregation at all. Each row represents an instance set and every column (starting at the second) represents the running time for each instance in that set.

The results show, not surprisingly, that the problem becomes intractable when the number of time periods become large. Planning for one year using a daily granularity proves itself intractable using the natural MIP model.

To analyze the effectiveness of the proposed disaggregation schemes, we ran the algorithm 1 for each instance with the following settings: the algorithm stops when 95% of the total freight can be accommodated with the installed capacity, when one of the MIPs or LPs cannot be solved in an hour, or when the total time reaches four hours.

Table 3 details the number of time aggregations in the model when the installed capacity is sufficient to accommodate 95% of the freight for those instances that did not terminated because of a time limit. Recall that

in each iteration two new breakpoints are introduced when using commodity defects and one new breakpoint is introduced when using time criticality to refine an aggregation.

Inst	Size (δ)	Size (χ)	Inst	Size (δ)	Size (χ)
1	5	6	10	13	22
2	9	16	11	17	43
3	10	21	12	13	15
4	3	8	13	11	30
5	7	11	14	5	2
6	9	14	15	7	16
7	11	22	16	-	-
8	9	11	17	5	7
9	5	15	18	-	3

Table 3. The number of breakpoints in the aggregation when 95% of the freight can be accommodated. A missing value indicates that the algorithm could not identify the desired network configuration within a reasonable amount of time.

The results are encouraging, but also show that more research and experimentation is needed. It is encouraging to see that for almost all instances it is possible in a few iterations, and thus with a very course aggregation, to identify a network configuration with installed capacity sufficient to accommodate 95% of the freight of the commodities.

However, these preliminary computational experiments also revealed that the convergence of the adaptive disaggregation procedure reduces dramatically in later iterations, when attempting to identify a network configuration with installed capacities sufficient to accommodate all the freight, thus leading to excessive run times. We could, however, obtain significant size reductions in every case. This is specially interesting when more time periods are considered. Since a significant part of the freight can be transported with the obtained solutions, and are therefore good quality bounds, these values can be used to hot-start a branch-and-cut procedure, improving its performance. Since these values are not expected to change significantly to be able to transport all the freight, further research would include the development of strategies to polish the obtained solutions to obtain the optimal, or at least good quality feasible solutions.

Another surprising observation, which is currently under further investigation, were the running times of the max flow solves, i.e., identifying the maximum amount of freight that can be accommodated with a certain network configuration. Even though the time-expanded network is large, we had expected faster run times. It may be necessary to switch from a general purpose LP solver for a purpose-built max flow solver.

5 CONCLUSIONS AND FINAL REMARKS

Dynamically adjusting time aggregations is a potentially powerful technique, especially for handling situations in which the natural optimization models get too large. Our preliminary computational experience with such approaches is encouraging, but also demonstrates that more research is necessary.

It is critically important to choose breakpoints wisely and perform as few iterations as possible. Therefore, further work would include the investigation of additional strategies to choose breakpoints to disaggregate at, of strategies that identify multiple breakpoints in each iteration, and use more information from the incumbent solutions.

REFERENCES

Ahmed, S., A. J. King, and G. Parija (2003, May). A Multi-Stage Stochastic Integer Programming Approach for Capacity Expansion under Uncertainty. *Journal of Global Optimization 26*(1), 3–24.

Floudas, C. A. and X. Lin (2004, October). Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Computers & Chemical Engineering 28*(11), 2109–2129.

Luss, H. (1982, September). Operations Research and Capacity Expansion Problems: A Survey. *Operations Research 30*(5), 907–947.

Newman, A. M. and M. Kuchta (2007, January). Using aggregation to optimize long-term production planning at an underground mine. *European Journal of Operational Research 176*(2), 1205–1218.

Ordóñez, F. and J. Zhao (2007, September). Robust capacity expansion of network flows. *Networks 50*(2), 136–145.