The congested multicommodity network design problem

Dimitris C. Paraskevopoulos

School of Management, University of Bath, Claverton Down, Bath, BA2 7AY, UK,

Sinan Gürel

Industrial Engineering Department,Middle East Technical University,06800 Ankara, Turkey

Tolga Bektaş

Southampton Business School, Centre for Operational Research, Management Science and Information Systems (CORMSIS), University of Southampton, Southampton, Highfield, SO17 1BJ, United Kingdom

Abstract

This paper studies a version of the fixed-charge multicommodity network design problem where in addition to the traditional costs of flow and design, congestion at nodes is explicitly considered. The problem is initially modeled as a nonlinear integer programming formulation and two solution approaches are proposed: (i) a reformulation of the problem as a mixed integer second order cone program to optimally solve the problem for small to medium scale problem instances, and (ii) an evolutionary algorithm using elements of iterated local search and scatter search to provide upper bounds. Extensive computational results on new benchmark problem instances and on real case data, are presented.

1. Introduction

Congestion is one of the causes for delay at freight hubs, e.g. yards, ports, or even cities. On 7 May 2012, a headline of a The New York Times article read "Freight Train Late? Blame Chicago", reporting that "Shippers complain that a load of freight can make its way from Los Angeles to Chicago in 48 hours, then take 30 hours to travel across the city. A recent trainload of sulfur took some 27 hours to pass through Chicago – an average speed of 1.13 miles per hour, or about a quarter the pace of many electric wheelchairs." [1] The article also claimed that the freight volume in the United States is projected to grow by at least 80% in the next 20 years which will have significant knock-on effects on delays. It is a well known fact that freight cars, in a rail network, spend most of their time in terminals or classification yards (Li et al., 2014). This is due to the fact that the same facility has to be used for consolidation and classification operations for a variety of vehicles carrying different types of freight. In these yards, cars usually go through the following

[1] http://www.nytimes.com/2012/05/08/us/chicago-train-congestion-slows-whole-country.html

operations: inspection, classification, assembly, accumulation and connection. As Fernandez *et al.* (2004) point out, the classification process constitutes the fundamental source of delay in the terminals, and this increases with the amount of classification, which is correlated with the number of cars to classify.

Congestion is prevalent not only in rail but in other transportation networks and modes as well, and has been the subject of recent research. For example, Tirachini *et al.* (2014) looked at the interplay of traffic congestion and bus crowding in public transport. By explicitly considering the social impact of congestion, the authors experimented with various variables of the system and found, among others, that an optimal frequency of the buses results from a the trade-off between the passenger crowd in the bus and the traffic congestion on the streets. The most common way of reducing the associtated social cost is by charging additional costs and preventing travelers from using particular transportation links (and/or nodes), thus reducing congestion (Yao *et al.* 2010, Laval *et al*. 2015). Fosgerau (2011) proved that a fast lane can replace congestion tolls at peak times, putting the focus on the balance between the capacity of the network and congestion pricing.

Traffic congestion is also linked with increased vehicle idling, acceleration and braking, which in turn increases engine related emissions. There is a rich literature on the environmental impacts of transportation and distribution logistics, with a particular focus on emissions (e.g., Demir *et al.*, 2010). Chen and Yang (2012) presented different toll schemes for minimizing both congestion and emissions in a bi-objective optimization approach. Franceschetti *et al.* (2013) looked at the impact of the time spent on a route on the total emissions. In particular, the objective function accounts for traffic congestion which, during peak hours, slows down the vehicles and increases emissions. The proposed model determines the optimum speed for a vehicle on each link of a route with an objective to minimize a broader objective function including emissions. Koç *et al.* (2014) studied the problem of routing of a heterogeneous fleet of vehicles using environmental objectives.

Designing and building a robust transportation network is a difficult and a multi-faceted decision problem of strategic importance. The fixed-charge multicommodity network design (MCND) model is extensively used to represent a wide range of planning and operation management problems in transportation, telecommunications, logistics and production-distribution. In its general form, the network design problem consists of designing a network on a given graph by selecting links to connect a set of nodes and to determine the amount of flow on each link such that the demand of each node for a set of commodities is satisfied. The objective is to minimize the total cost of establishing the links and flows. This basic variant is usually referred to as the *uncapacitated network design problem*, which has extensions incorporating additional restrictions, such as capacity limits on the amount of demand that may be transported on the links. Interested readers on the problem may consult the surveys by Magnanti and Wong (1986), Minoux (1986) and Crainic (2000).

In this paper, we model and study the fixed-charge MCND problem (MCNDP) where congestion at nodes (e.g., yards) is explicitly taken into consideration. This problem, named as the *congested multicommodity network design problem* (cMCNDP), is what we believe to be one of the first to incorporate congestion into this particular setting. Our primary motivation stems from the application of this model in planning freight rail transportation systems and to be able to explicitly capture congestion in the respective models and solution methods. The problem considered here also allows for capacity expansion (Liu *et al.*, 2008) for reducing congestion. The contribution of this study is two-fold: (a) to describe a reformulation of problem as a mixed integer second order

cone program (MISOCP) which is used to optimally solve the problem for small to medium scale instances, (b) to present an evolutionary heuristic using iterated local search and scatter search.

The rest of the paper is structured as follows. Section 2 provides background on modeling delay. Section 3 formally describes the problem and provides the notation as well as a small numerical example. Section 4 describes an integer programming formulation and the MISOCP reformulation. Section 5 describes the evolutionary algorithm and all of its components. Section 6 presents results of extensive computational experiments on a large set of augmented benchmark instances and on real case data. Conclusions are given in Section 7.

2. Modeling Delay

There are various approaches to model yard delays, simulation and queueing models being two of them. The latter are more attractive in the sense that they can be used to derive analytical expressions and are easy to incorporate in tactical decision models. Crainic (2005) mentions that "most time-related functions are built to reflect the increasingly larger delays that result when facilities of limited capacity must serve a growing volume of traffic. Such congestion functions are typically derived from engineering procedures and queuing models".

Various analytical expressions have been proposed in the literature to model yard delays. Petersen (1977a, b) proposed several models for different components of the classification process and studied models that are based on the physical characteristics of the yard. Later, Turnquist and Daskin (1982) proposed a batch arrival queuing model for the same operation. These two approaches are based on individual characteristics of the yard. Crainic *et al.* (1984) argued that such precise data may be difficult to obtain and may not be necessary within a tactical level planning perspective and proposed two analytical formulas to calculate classification delays, both based on the $M/M/1$ queueing model. The first and the one relevant to our discussion can be used to calculate the mean classification delay at a yard and is as follows:

$$\frac{Tt}{T - tf}, \tag{1}$$

where T denotes the length of the planning period, t is the mean service time for a yard and f is the total amount of traffic to be classified at this yard. Fernandez *et al.* (2004) proposed to calculate classification delays, not based on trains, but based on individual freight cars. The authors argued that such an approach will result in a more precise and reliable modelling of classification delays. The expression they propose instead to calculate the average classification delay for a freight car in a yard is the following:

$$F + \beta \left(\frac{f}{S}\right)^{\alpha}, \tag{2}$$

where F is the classification delay for a freight car under free flow conditions, f denotes the amount of freight cars to be classified in the yard during the period of analysis, S is the classification capacity of the yard over the time of analysis, and β, α are the calibration parameters. Since this function measures the average classification delay for a freight car in a particular yard, the total delay in the yard with a flow of f freight cars will be,

$$Ff + \beta \frac{f^{\alpha+1}}{S^{\alpha}}. \tag{3}$$

Expressions of type (3) are based on the delay functions initially proposed by the US Bureau of Public Roads (see Gürel, 2011, for a related discussion). In the context of the MCND, the difficulty in incorporating this type of functions is that they give way to nonlinear integer programming formulations of the problem as will be shown below.

There is some literature on nonlinear multi-commodity network problems. The body of literature on the "flow" side includes the earlier work by Gerla (1973) and later on by Mahey *et al.* (2001). An excellent survey of the convex multicommodity network flow problem is provided in Ouorou (2000) where the structure of the continuous formulations presented therein is similar to the formulation described here, with the difference being that the latter is discrete. Recently, Gürel (2011) has presented ways of reformulating network flow problems with convex congestion functions leading to an efficient way of solving the resulting nonlinear models. On the "design" side, Crainic and Rousseau (1986) considered a nonlinear, mixed integer, multimodal, multicommodity network flow problem and proposed a solution algorithm that is a combination of a heuristic and a convex network optimization procedure. The latter procedure is based on column generation and descent techniques. Croxton *et al.* (2003) considered a multicommodity network flow problem with piecewise linear costs, described structural results for various formulations of the problem and presented the results of extensive computational experiments carried out on these formulations. A multicommodity network design problem with discrete node costs is studied in Belotti *et al.* (2006), where node costs are stepwise functions of the facilities installed into the nodes. The authors proposed, for the solution of this problem, a branch-and-cut algorithm based on two families of valid inequalities. A variant of the MCND where capacity constraints are penalized was presented by Bektaş *et al.* (2010), which was modeled as a nonlinear MCND formulation for which the authors described Lagrangean-based solution algorithms. Mathematical models for routing problems incorporating delay constraints were presented in Ben-Ameur and Ouorou (2006) and those featuring "on/off" constraints appear in Hijazi *et al.* (2012). A more recent and relevant work is by Frangioni *et al.* (2015), who studied the single-flow and single-path routing problem with constraints on delay. The authors showed that the problem can be formulated as a convex mixed-integer nonlinear optimization problem, which can then be represented as second-order cone models and solved by efficient general-purpose solvers.

There also exists work on incorporating congestion into other types of design problems. For example, Elhedli and Hu (2005) looked at hub-and-spoke design where a congestion function of a simple, but convex (quadratic) nature, was incorporated into the existing models. Elhedli and Hu (2005) described methods based on Lagrangean relaxation to solve the resulting nonlinear models. Later, Elhedli and Wu (2010) extended this problem in which capacity selection was considered as an extra layer of decision. This work addressed the hub-and-spoke system as a network of $M/M/1$ queues and proposed a Lagrangean heuristic for its solution. A more recent work looking at congestion within telecommunication networks is by Miranda *et al.* (2011), who presented a network design model for the problem incorporating a nonlinear convex function integrating capacity expansion and congestion function. The authors described an algorithm based on Generalized Benders Decomposition to solve the nonlinear network flow problem arising when the design variables in the model are fixed.

Recently, Khaled et al. (2015) presented a mathematical model for addressing disruptions in rail networks. The authors explicitly considered delays that occur on links and yards as a result of disruption, and looked at re-routing of trains onto links and nodes adjacent to the location suffering from the disruption. Fan *et al.* (2010, 2012) examined congestion at ports and developed a network flow model in an intermodal setting. Fan *et al.* (2012) stated that congestion is dominant

in ports, traffic is diverted into different routes most of the time, and showed that expansion of capacity would dramatically reduce congestion costs and waiting times.

3. Problem Description

The cMCNDP is defined on a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ where $\mathcal{N}$ is the set of nodes and $\mathcal{A}$ is the set of arcs. For each node $i \in \mathcal{N}$, we define the sets $\mathcal{N}_i^+ = \{j \in \mathcal{N} | (i,j) \in \mathcal{A}\}$ and $\mathcal{N}_i^- = \{j \in \mathcal{N} | (j,i) \in \mathcal{A}\}$. For each open arc in the network, there is a fixed charge denoted by f_{ij}. Note that, the "open" arcs are the arcs that are selected to accommodate commodity flow. There exists a set of commodities denoted by $\mathcal{P}$. Each commodity has one origin $o(p)$ and one destination $d(p)$, and the quantity of commodity p that is to be sent from $o(p)$ to $d(p)$ is denoted by w^p. For convenience, we define the parameter d_i^p for each $i \in \mathcal{N}$ that equals w^p if $i = o(p)$, $-w^p$ if $i = d(p)$, and 0 otherwise. If a commodity has more than one origin or destination, this can be modelled by splitting the commodity into several commodities, each with a single origin and destination (see Holmberg and Yuan, 2000). We denote by d_{ij}^p the unit cost of routing the demand for commodity p over arc (i,j). Each arc (i,j) in the network has a capacity u_{ij}. For any yard $i \in \mathcal{N}$, let $c_i^0 > 0$ denote the yard's initial capacity, e_i denote the cost of upgrading/expanding the capacity of yard, and let $c_i^\delta > 0$ denote the upgrade capacity. The assumption behind only one level capacity upgrade, as opposed to multiple levels, is mainly dictated by current practice found in railyards. One example is provided in Petersen (1977b) for a single-ended yard with an initial capacity of seven classification tracks, where the expansion is achieved by adding three more classification tracks. A more practical example is from the MacMillan Yard in Toronto, which has a dual hump with two tracks (TSB, 2003). Prior experience of one of the authors is that the hump it is mostly used in a single mode. Nevertheless, the existence of the dual lead tracks still provides some flexibility in the in the way of a one-step capacity expansion (Crainic and Bektaş, 2006; Bektaş et al., 2009).

The function we use to measure congestion is (3) as described by Fernandez *et al.* (2004). The free-flow congestion at yard $i \in \mathcal{N}$ is denoted by F_i, and the unit congestion cost is shown by D_i. The cMCNDP consists of finding flows for each type of commodity from its origin to its destination by activating suitable arcs and performing capacity upgrades in network $\mathcal{G}$ so as to minimize a total cost function and obey certain constraints. The total cost function is composed of four components: (i) cost of routing commodities, (ii) cost of activating arcs, (iii) cost of upgrading yard capacities, and (iv) cost of congestion in each yard. The constraints pertain to limits on the amount of commodities that flows on each arc due to arc capacities and the total amount of commodities flowing into each yard due to yard capacities.

3.1. A motivating example

This section provides an example on a small-scale instance to show the effect of taking congestion into account in multicommodity network design. Assume a five node instance, as given in Figure 1, where all links have unit capacities and unit fixed-charge costs. The two links into and the two out of node 3 have unit flow costs, links 1–4 and 2–5 have unit flow cost of 10 each. There are two commodities p_1 and p_2 to be shipped, with $w_{p_1} = 1$, $o(p_1) = 1$, $d(p_1) = 4$ and $w_{p_2} = 1$, $o(p_2) = 2$, $d(p_2) = 5$. We assume that node 3 in the network is a transhipment point (e.g., a rail-yard) with a delay cost arbitrarily set equal to $D_3 = 5$ given that the total congestion cost increases linearly with this parameter. The capacity of node 3 has initially been set equal to the total amount of commodity flows in the network, which is $c_3^0 = w_{p_1} + w_{p_2} = 2$. All other nodes are origin or

destination points with $D_i = 0$, $i = 1, 2, 4, 5$ as they are either origin or destination nodes, for which neither capacity nor congestion is relevant for the purposes of this numerical example. We also assume the following initial settings on parameters $\alpha = 3$, $\beta = 3$ and $F = 1$, but the effect of changing the former two parameters against the capacity of node 3 will be shown later.

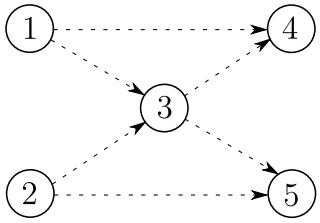
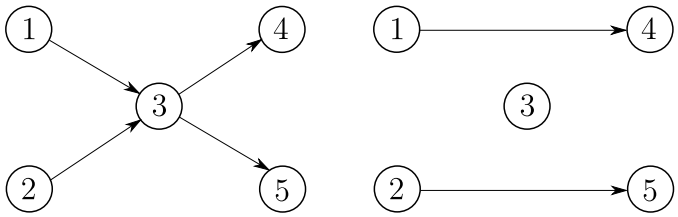


Figure 1: A five node MCND instance

An MCND solution on the sample 5-node instance is given in Figure 2(a) where both commodities go through node 3 with a total cost of 8 units consisting only of flow and fixed-charge costs. However, if one calculates the "hidden" cost of congestion through function (3), the overall cost rises to $8 + 5(2 + 3(2^4/2^3)) = 48$ units. Alternatively, a solution minimizing the total cost including congestion is given in Figure 2(b) wherein the commodities are shipped from their origins to their destinations as direct deliveries. The total cost of this solution is 22 units.



(a) G_1 (b) G_2

Figure 2: Two solutions on the sample instance: (a) minimizing fixed and flow costs and (b) minimizing congestion at node 3

The two solutions shown in Figure 2 are two extreme cases one might encounter. The solutions obtained will certainly vary with the parameters chosen for the congestion function but the example shown here serves to illustrate cases where incorporating a congestion function into this problem might significantly change the structure of the solution on the same instance. To provide more information as to how the total cost of the solution changes with varying values of some of its input parameters, we present Figure 3.

Figure 3 shows the shape of the total solution cost comprising fixed, variable and delay costs in the vertical axis against varying values of initial capacity c_3^0 in the horizontal axis. The total cost is shown by Cost (α, β) as a function of the two calibration parameters. It is clear from the figure that the total cost starts to level off when c_3^0 exceeds a certain value, which, in this case is between 5 and 6. In fact, both solutions shown in Figures 2a and 2b become alternative optima for Cost (3,3) when $c_3^0 = 4$, and for values $c_3^0 > 4$, the former is preferred over the latter. This figure also shows that, for all the three cases, there is a significant drop in total cost when the initial capacity

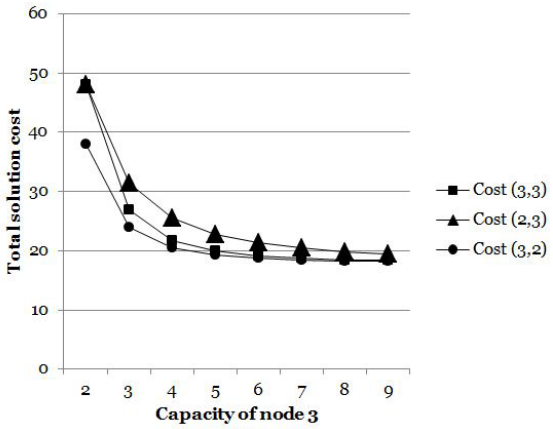


Figure 3: Congestion function with different values of α and β

$c_3^0 = 2$ is increased by a single unit.

The following section presents the development of a mathematical model for the cMCNDP.

4. Mathematical Formulations

In this section, a notation of the problem is summarised followed by the mathematical formulation. The following decision variables are defined to model the cMCND. Let $x_{ij}^p \geq 0$ denote the amount of flow of commodity $p \in \mathcal{P}$ on arc $(i,j) \in \mathcal{A}$. The binary variables of the model are given below:

$$y_{ij} = \begin{cases} 1 & \text{arc } (i,j) \in \mathcal{A} \text{ is used.} \\ 0 & \text{otherwise,} \end{cases} \tag{4}$$

$$z_i = \begin{cases} 1 & \text{if node } i \in \mathcal{N} \text{ is upgraded} \\ 0 & \text{otherwise.} \end{cases} \tag{5}$$

The following sets are used in the model:

- $\mathcal{A}$: set of arcs,
- $\mathcal{P}$: set of commodities,
- $\mathcal{N}$: set of nodes.

The list of parameters used in the model are as follows:

- e_i: fixed cost of upgrading the node $i \in \mathcal{N}$,
- f_{ij}: fixed cost of establishing (opening) an arc $(i,j) \in \mathcal{A}$,
- d_{ij}^p: the unit cost of routing the demand for commodity $p \in \mathcal{P}$ over arc $(i,j) \in \mathcal{A}$,
- c_i^δ: the upgrade capacity of node $i \in \mathcal{N}$,

- c_i^0: the initial capacity of node $i \in \mathcal{N}$,
- w_p: the quantity of commodity $p \in \mathcal{P}$ to be shipped,
- u_{ij}: the capacity of an arc $(i,j) \in \mathcal{A}$,
- D_i: the delay cost of node $i \in \mathcal{N}$,
- F_i: the free flow classification delay for node $i \in \mathcal{N}$,
- α, β: calibration parameters,
- v_i: the total amount of flow into node $i \in \mathcal{N}$.

Given the notation above, a mathematical programming formulation for the problem is presented below:

$$\mathcal{M} \quad \text{Minimize} \sum_{(i,j)\in\mathcal{A}} f_{ij}y_{ij} + \sum_{(i,j)\in\mathcal{A}} \sum_{p\in\mathcal{P}} d_{ij}^p x_{ij}^p + \sum_{i\in\mathcal{N}} e_i z_i + \sum_{i\in\mathcal{N}} g_i(x,z) \tag{6}$$

$$\text{subject to} \sum_{j\in\mathcal{N}_i^+} x_{ij}^p - \sum_{j\in\mathcal{N}_i^-} x_{ji}^p = d_i^p \qquad \forall i \in \mathcal{N}, p \in \mathcal{P} \tag{7}$$

$$x_{ij}^p \le w^p y_{ij} \qquad \forall (i,j) \in \mathcal{A}, p \in \mathcal{P} \tag{8}$$

$$\sum_{p\in\mathcal{P}} x_{ij}^p \le u_{ij} y_{ij} \qquad \forall (i,j) \in \mathcal{A}, \tag{9}$$

$$\sum_{j\in\mathcal{N}^-} \sum_{p\in\mathcal{P}} x_{ji}^p \le c_i^0 + c_i^\delta z_i \qquad \forall i \in \mathcal{N}, \tag{10}$$

$$y_{ij} \in \{0,1\} \qquad \forall (i,j) \in \mathcal{A} \tag{11}$$

$$x_{ij}^p \ge 0 \qquad \forall (i,j) \in \mathcal{A}, p \in \mathcal{P} \tag{12}$$

$$z_i \in \{0,1\} \qquad \forall i \in \mathcal{N}. \tag{13}$$

In formulation $\mathcal{M}$, the objective function represents the total cost of design, routing and capacity augmentation and cost of congestion. The function $g_i(x,z)$ that appears in the last term of the objective function models congestion and is expressed as follows,

$$g_i(x,z) = D_i \left(F_i \sum_{j\in\mathcal{N}^-} \sum_{p\in\mathcal{P}} x_{ji}^p + \beta \frac{\left(\sum_{j\in\mathcal{N}^-} \sum_{p\in\mathcal{P}} x_{ji}^p \right)^{\alpha+1}}{(c_i^0 + c_i^\delta z_i)^\alpha} \right), \tag{14}$$

where α and β are calibration parameters.

In this formulation, (7) are the flow conservation constraints which ensure that the demands are satisfied for each node. Constraints (8) make sure that the flow of any commodity on an arc is zero when that arc is not selected. Constraints (9) imply that the amount of flow on an arc can be at most equal to the capacity of the arc. Finally, constraints (10) limit the total inflow of the node i by

its initial (i.e., $z_i = 0$) or extended (i.e., $z_i = 1$) capacity. Integrality and nonnegativity restrictions on the decision variables are given by (11)–(13). Model $\mathcal{M}$ has the structure of a nonlinear, mixed integer, multicommodity network design problem including congestion effects and upgrading decisions at nodes.

The following section will present a conic mixed integer programming formulation. The latter requires the following notation change in the problem formulation. We first introduce an additional variable, $v_i \geq 0$, which denotes the total amount of flow into node $i \in N$. The new variable is expressed, in mathematical terms, as follows:

$$v_i = \sum_{j \in \mathcal{N}^-} \sum_{p \in \mathcal{P}} x_{ji}^p \qquad \forall i \in \mathcal{N}. \tag{15}$$

Under the new variable definition, the congestion function becomes

$$g_i(v, z) = D_i \left(F_i v_i + \beta \frac{(v_i)^{\alpha+1}}{(c_i^0 + c_i^\delta z_i)^\alpha} \right).$$

In the following sections, we describe two methods for the problem, namely a conic mixed integer programming reformulation of $\mathcal{M}$, and an evolutionary algorithm, in the given order.

4.1. MISOCP reformulation

Formulation $\mathcal{M}$ given in Section 4 is a mixed integer nonlinear programming problem due to the second term $g(v, z)$ appearing in the objective function. For the purposes of the reformulation, we represent the nonlinear part of $g(v, z)$ by $g^{nl}(v, z)$ as shown below (indices i are dropped for the sake of simplifying the exposition).

$$g^{nl}(v, z) = D\beta \frac{(v)^{\alpha+1}}{(c^0 + c^\delta z)^\alpha}.$$

The function $g^{nl}(v, z)$ gives rise to two congestion cost functions, $g^{nl}(v, 0)$ and $g^{nl}(v, 1)$, corresponding to capacity levels c^0 (i.e., $z = 0$) and $c^0 + c^\delta$ (i.e., $z = 1$), respectively. Figure 4 illustrates these two cost functions.

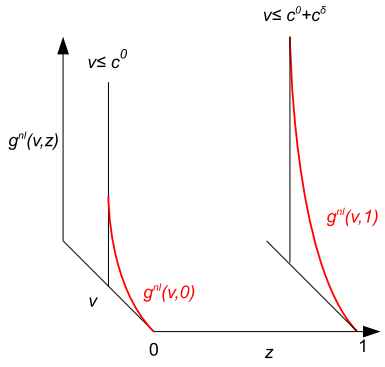


Figure 4: Congestion Costs at Different Capacity Levels

Obviously, for a fixed value of z such that $z > -\frac{c^0}{c^\delta}$, $g^{nl}(v, z)$ reduces to a convex function of v

given that $\alpha > 1$. When z is not fixed $g^{nl}(v, z)$ defines a surface. Figure 5 presents the surface defined by $g^{nl}(v, z)$ when restricted by the constraint $v \leq c_0 + \delta z$.

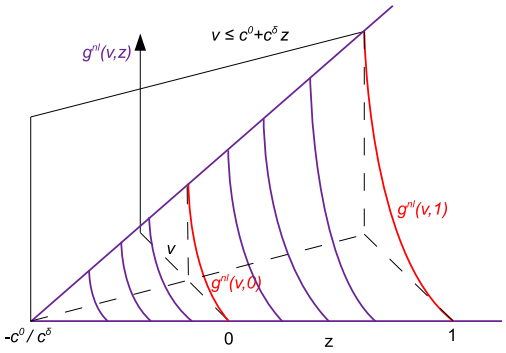


Figure 5: Surface Defined by $g^{nl}(v, z)$ and $v \leq c^0 + c^\delta z$

In this section, we show that we can reformulate $\mathcal{M}$ as a conic mixed integer programming formulation, for which the following result will be used.

Proposition 1. *$g^{nl}(v, z)$ is a Second Order Cone Programming (SOCP)-representable function.*

Proof. A function is SOCP-representable if its epigraph is so. Hence, we consider the following inequality

$$D_i \beta \frac{(v_i)^{\alpha+1}}{(c_i^0 + c_i^\delta z_i)^\alpha} \leq t \qquad \forall i \in \mathcal{N}. \tag{16}$$

Inequality (16) can be equivalently written as

$$D_i \beta v_i^{\alpha+1} \leq t \cdot (c_i^0 + c_i^\delta z_i)^\alpha \qquad \forall i \in \mathcal{N}, \tag{17}$$

which is of the form

$$r^{2^l} \leq s_1 s_2 \cdots s_{2^l}, \tag{18}$$

for $r, s_1, \ldots, s_{2^l} \geq 0$. Inequalities of form (18) are SOCP-representable (Ben-Tal and Nemirovski, 2001). We can express inequality (18) by using $O(2^l)$ variables and $O(2^l)$ hyperbolic inequalities of the form

$$u^2 \leq v_1 v_2, \; u, v_1, v_2 \geq 0. \tag{19}$$

Each hyperbolic inequality (19) can be written as a second-order conic inequality

$$\|(2u, v_1 - v_2)\| \leq v_1 + v_2. \tag{20}$$

Thus, $g^{nl}(v, z)$ can be represented via SOCP constraints. □

Gürel (2011) has shown that a widely used form of congestion functions, which is a class of convex power functions, can be represented via second-order conic inequalities. The same author has also shown that, due to the polyhedral characteristics these functions have, they can be efficiently computed in network flow problems. We observe that $g^{nl}(v, z)$ fall into this class and hence we reformulate our problem as a conic formulation using the SOCP representation of $g^{nl}(v, z)$.

As given in the proof of Proposition 1, when reformulating the problem, we first replace $g^{nl}(v,z)$ with an auxiliary variable $t \geq 0$, following which we include the SOCP representation of inequality (16) in our formulation. Example 1 shows how the SOCP representation for an example function $g^{nl}(v,z)$ can be obtained.

Example 1. Consider a congestion cost function in which $\alpha = \frac{3}{2}$, $D_i\beta = 1$, $c_i^0 = 2$ and $c_i^\delta = 3$. Then, inequality (16) becomes $v^{5/2} \leq t(2+3z)^{3/2}$, which can be equivalently written as $v^{5/2} \leq ty^{3/2}$ and $y = 2 + 3z$. The former is equivalent to $v^5 \leq t^2y^3$, which can be rewritten as,

$$v^8 \leq t^2y^3v^3, \tag{21}$$

which is a special case of inequality (18).

In order to obtain the SOCP representation of inequality (21), we follow the construction of Alizadeh and Goldfarb (2003). In particular, we build the inequality (21) by using a binary tree with leaf nodes for $\{t, t^2, t^4, \ldots\}$, $\{y, y^2, y^4, \ldots\}$, and $\{v, v^2, v^4, \ldots\}$. Then, using the binary representation of the exponents of t^2, v^3, and y^3 on the right hand side of inequality (21) we form the leaves of the construction tree as shown in Figure 6. Note that, in order to express an integer exponent, we only use powers of 2 as the leaves of the binary tree. Each non-leaf node of the binary tree represents a new hyperbolic inequality (19) and the new variable introduced.

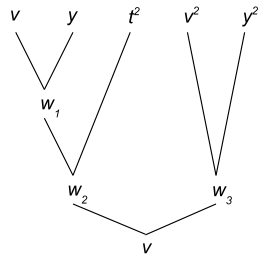


Figure 6: Binary Representation Tree for Example 1

Using the construction above, we can express inequality (21) equivalently through the following set of hyperbolic constraints: $w_1^2 \leq vy$, $w_2^2 \leq w_1 t$, $w_1 \geq 0$, $w_3^2 \leq vy$, $v^2 \leq w_2 w_3$, $w_2 \geq 0$ and $w_3 \geq 0$. One can easily check this equivalence by squaring the last inequality twice to achieve v^8 on its left-hand side and then substituting w_i's on the right-hand side appropriately. The hyperbolic constraints are then written in SOCP form as: $\|(2w_1, v-y)\| \leq v+y, \|(2w_2, w_1-t)\| \leq w_1 + y, \|(2w_3, v-y)\| \leq v+y, \|(2v, w_2-w_3)\| \leq w_2+w_3$.

As a result, $\mathcal{M}$ can be reformulated as follows,

$$\mathcal{M}_{MISOCP} \qquad \text{Minimize} \sum_{(i,j)\in\mathcal{A}} f_{ij}y_{ij} + \sum_{(i,j)\in\mathcal{A}} \sum_{p\in\mathcal{P}} d_{ij}^p x_{ij}^p + \sum_{i\in\mathcal{N}} e_i z_i + \sum_{i\in\mathcal{N}} D_i F_i v_i + D_i\beta t_i, \tag{22}$$

subject to (7)–(9), (11)–(15), and

$$\text{SOCP representation of } v_i^{\alpha+1} \leq t_i(c_i^0 + c_i^\delta z_i)^\alpha \qquad \forall i \in \mathcal{N}. \tag{23}$$

5. Solution Procedure

This section describes an evolutionary algorithm for the cMCNDP. Recently proposed heuristic methodologies for the traditional MCNDP use a trajectory-based algorithm (Ghamlouche *et al.* 2003, Crainic *et al.* 2006) or an evolutionary framework (Ghamlouche *et al.* 2004, Alvarez *et al.* 2005) to select the arcs of the network, following which an off-the-shelf optimizer is used to solve the linear programming flow problem on the network configuration. In this paper, a similar approach is followed, although in this case the flow problem that arises within the algorithm is nonlinear. The details of the algorithm and the way in which the nonlinear subproblem is handled is explained below.

The solution methodology is an adaptation of the Cycle-based Evolutionary Algorithm (CEA) proposed by Paraskevopoulos *et al.* (2014), in which additional decisions and aspects that the cMCNDP introduces are taken into account. In particular, a tailor-made cMCNDP evolutionary algorithm is proposed wherein (a) the solution recombination takes into account both node upgrading decisions and links establishments from the parent solutions to produce offspring (b) an efficient perturbation strategy is used that uses long term memory to guide the search towards unexplored regions of the solution space, (c) and the local search employs new neighbourhood operators that explicitly consider congestion at nodes.

Algorithm 1: Evolutionary Algorithm

Input: λ (initial population size), μ (Reference Set size), ψ (number of local search iterations without an improvement), κ (Candidate Set size), ϑ_{max} (number of SOCP solver calls within local search without an improvement)
Output: Reference Set(R), $s_{best} \in R$
1. Initialization phase
$R \leftarrow$ ConstructionHeur(λ, μ);
while *termination conditions* **do**
  2. Scatter Search phase
  $M \leftarrow \emptyset$, $M \leftarrow$ SolutionCombination(κ, μ);
  3. Education phase
  for *individual* s *of* M **do**
    $s' \leftarrow$ ILS$(s, \psi, \vartheta_{max})$;
    UpdateRefSet(R, s');

In summary, the evolutionary algorithm is based on principles of Scatter Search (SS) and integrates an Iterated Local Search (ILS) as an improvement, or an "education" method. Within the algorithm, the basic three-phase SS scheme is followed, namely (i) initialization, (ii) scatter search to produce offspring, (iii) education phase using the ILS. These steps will be described in greater detail below. A pseudecode of the overall framework is presented in Algorithm 1.

Figure 7 shows the flow chart of the proposed evolutionary algorithm. Within the algorithm, Scatter Search comprises the Solution Combination method and the population of offspring, whereas Iterated Local Search, described in Section 5.3.2, includes Local Search and Perturbation. Evaluation of the solutions is performed by the Reference Set Update procedure, described in Section 5.2. In the following, emphasis is given on the new aspects of the proposed methodology rather than the generic framework, the details of which can be found in Paraskevopoulos *et al.* (2014).

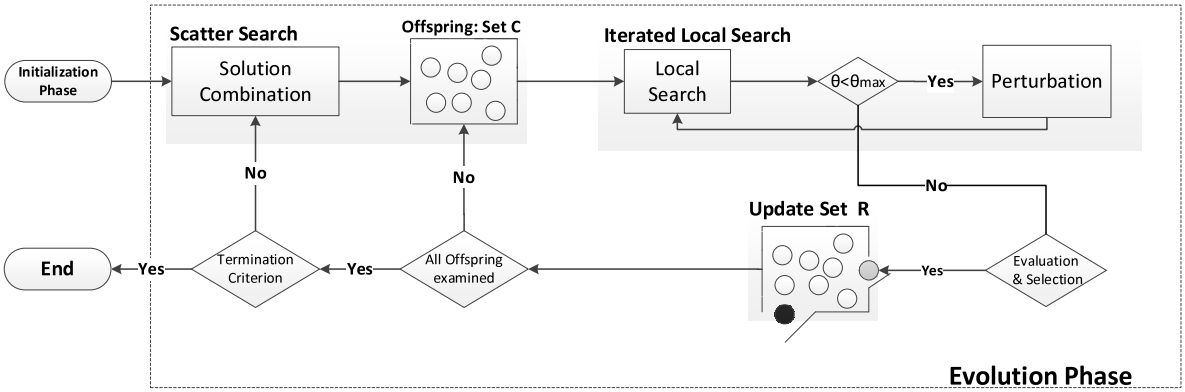


Figure 7: The flow chart of the Evolutionary Algorithm

5.1. Initialization phase

The *initialization phase* uses a constructive heuristic to initialize a pool of μ diversified and good-quality solutions, which are hosted in the Reference Set R. A serial construction procedure routes a particular commodity at each iteration, and finds the best path for that commodity by considering options of splitting or routing the total amount as a whole. The paths are found by using Dijkstra's shortest-path algorithm. A greedy function is used that considers the fixed and flow cost of the arcs, as well as the congestion and upgrading costs on the nodes, and ensures that arcs and nodes capacities are always satisfied. Given the set of open arcs and the upgraded nodes, i.e., the 0–1 decision variables y_{ij} and z_i fixed in formulation $\mathcal{M}_{MISOCP}$, the resulting flow problem is a SOCP formulation which is solved by a SOCP solver. Its solution provides an optimal flow of commodities on the network.

This algorithm also handles the decisions on whether to upgrade a node or not internally as follows. At any iteration of the Dijkstra algorithm, let the end node of a given arc be a candidate that is to be assigned a permanent label. Then, two congestion costs are calculated for this node; one that assumes a capacity upgrade on the node, and one that does not. Then, the procedure will assign a permanent label to the node that has the least total cost, including both the arcs costs and the congestion costs at nodes. If at a particular iteration a node is chosen to be upgraded, it will remain upgraded until the construction heuristic terminates and a complete solution is derived.

5.2. Reference Set Updating Criteria

There are certain criteria according to which the Reference Set R is updated with new solutions. The goal is to maintain a balance between quality and diversity and to avert premature convergence. A new solution is inserted into R if it is better than the ever best found in terms of the total cost, or if it increases the average dissimilarity of the R and its cost is better than the worst-cost solution in R. In both cases, the worst cost solution in R is removed and the new solution is inserted. To calculate the dissimilarity of the R, the current solution is temporarily inserted into R and the worst-cost solution in R is temporarily removed. The distance of all pairs of solutions is estimated by the Hamming distance, which is computed by considering both the arcs of solutions and the upgraded nodes, and is given by the formula below:

$$H(s,s') = \sum_{(i,j)\in\mathcal{A}} |y_{ij}^{s} - y_{ij}^{s'}| + \sum_{i\in\mathcal{N}} |z_i^{s} - z_i^{s'}|, \tag{24}$$

where, for a given solution s, y_{ij}^s is a binary variable equal to 1 if arc $(i,j) \in \mathcal{A}$ is open, or 0 if not. Similarly, z_i^s is a binary variable equal to 1 if node $i \in \mathcal{N}$ is upgraded, or 0 if not.

5.3. Scatter Search Phase

In the scatter search phase, R is evolved via an efficient solution combination method and an improvement method, namely the ILS. A subset generation method randomly selects κ solutions from R repeatedly, and μ offspring are produced based on scatter search principles. ILS attempts to improve the quality of each offspring, before the latter can be inserted into R according to the elitist update criteria described in Section 5.2.

5.3.1. Solution Combination method

The proposed solution combination method combines the solution elements of κ solutions selected from R to form the Candidate Set (C). These elements are both the arcs and nodes of the solutions. Each arc and each node status, i.e., open (or closed) and upgraded (or not upgraded) respectively, are assigned a value of preference. To calculate this value of preference κ weights are introduced, as many as the number of solutions in C. If an arc (or node) is found open/closed (or upgraded/not upgraded) in more than one solution, the value of preference of its status is calculated as the sum of the respective weights. A voting procedure is incorporated to determine which status is the dominant for each arc and for each node, according to the formulae given below:

$$\textit{Arc preferences:} \qquad Op_{ij} = \sum_{s \in C} \frac{y_{ij}^s}{f(s) + \gamma hits(s)} \qquad Cl_{ij} = \sum_{s \in C} \frac{1 - y_{ij}^s}{f(s) + \gamma hits(s)} \qquad \forall (i,j) \in \mathcal{A}, \tag{25}$$

$$\textit{Node preferences:} \qquad Up_i = \sum_{s \in C} \frac{z_i^s}{f(s) + \gamma hits(s)} \qquad nUp_i = \sum_{s \in C} \frac{1 - z_i^s}{f(s) + \gamma hits(s)} \qquad \forall i \in \mathcal{N}, \tag{26}$$

where γ is a positive normalization parameter, i.e., average cost of an arc in the best so far solution found, Op_{ij} and Cl_{ij} are the scores for the open and closed status, respectively, for the arc (i,j), and Up_i and nUp_i are the scores for the status of the node i (upgraded or not). The first component $f(s)$ in the denominators of (25) and (26) is the cost of the solution s. The second component denotes the number of times a particular solution s has participated in the recombination process. The latter is used for diversification purposes to prevent the recombination process from choosing frequently selected parents. The status with the higher value dominates, both for the arcs and the nodes.

The output of the above procedure is a preferable status for each and every arc and node. To build a feasible solution, one should determine also the commodities flows for each link, which is done through solving the associated non-linear programming flow problem as described above.

Due to the nature of the recombination process, it might happen that the set of the open arcs and upgraded nodes imply an infeasible flow problem. To restore feasibility, a reconstruction mechanism, similar to the one described in the initialization phase, is implemented. The goal of this mechanism is to produce a feasible solution which maintains as many of the open arcs and upgraded nodes as the recombination process suggests. Towards this end, the open arcs obtained by the SS are assigned very low costs, which forces the construction heuristic to choose them. For example, if arc (i,j) has a preferred status open, we temporarily amend the fixed cost to f_{ij}/L and

the flow cost to c_{ij}/L, where L is a large constant. The preferred status for the nodes is assigned in a subsequent step, where given the flows the upgrading decision might be made to preserve the capacity constraints at nodes. The latter overcomes the preferred status in case it is non-upgrade, at the expense of preserving the capacity constraints at nodes. Towards the other end, the reconstruction mechanism also incorporates a final check to see if there are any cost savings by down-grading any of the upgraded nodes, provided commodity flows still remain feasible.

5.3.2. Education phase

The offspring produced by the evolution phase are improved using ILS, applied to each individual offspring. ILS has two main components; a local search and a perturbation strategy. The proposed local search introduces new neighbourhood structures and uses short term memory to enable the escape from local optima. The perturbation strategy partially modifies the current solution according to information gathered during the search. The components of the ILS algorithm are shown in Algorithm 2.

Algorithm 2: Iterated Local Search

Input: s (current offspring), ψ (number of local search iterations without an improvement), ϑ_{max} (number of SOCP solver calls without an improvement)
Output: s
$\vartheta \leftarrow 0$,$\vec{h} \leftarrow 0$;
while $\vartheta < \vartheta_{max}$ **do**
  $\vec{r} \leftarrow 0$;
  $s' \leftarrow s$
  $count = 1$
  while $count < \psi$ **do**
    $N(s) \leftarrow$ Neighbourhood Evaluation$(s, \vec{r})$
    $s \leftarrow \min_{s'' \in N(s)} f(s'')$
    Update Inefficient Chains(s'')
    Update Memory Structures($s'', \vec{r}, \vec{h}$)
    if $\{f(s) < f(s')\}$ **then**
      $count = 1$
      $s' \leftarrow s$
    else
      $count = count + 1$
  $s' \leftarrow$ SOCPsolver(s');
  if $f(s') > f(s)$ **then**
    $s^* \leftarrow$ Perturbation(s',$\vec{h}$)
    $s \leftarrow s^*; \vartheta \leftarrow \vartheta + 1;$
  else
    $\vartheta \leftarrow 0; s \leftarrow s';$

In Algorithm 2, the *Update Inefficient Chains* function creates new inefficient chains, following the application of the local search operator is applied on solution s' (see Section 5.3.3 for details). The *Update Memory Structures* function then updates the memory structures $\vec{r}$ and $\vec{h}$ with information relevant to previously selected moves and neighbours, to enable diversification. The Neighbourhood Evaluation procedure that appears in Algorithm 2 is shown in Algorithm 3.

Algorithm 3: Neighbourhood Evaluation

Input: s (current solution), M a large number
Output: s'' (best neighbour)
$\min = M$;
for *All inefficient chains k of s and for all combinations of nodes i, j in k* **do**
  $\mathcal{P}_I \leftarrow$ ListCommodities(k, i, j);
  while *$\mathcal{P}_I$ is not empty* **do**
    if *isItFeasible($k, i, j, \mathcal{P}_I$)* **then**
      $s^* \leftarrow$ GenerateNeighbour($k, i, j, \mathcal{P}_I$);
    else
      RemoveFirstCommodity($\mathcal{P}_I$);Continue;
    if $\Delta f_{\text{move}}(s, s^*) < \min$ **then**
      $s'' \leftarrow s^*$; $\min = \Delta \text{f}_{\text{move}}(s, s^*)$;
    RemoveFirstCommodity($\mathcal{P}_I$);

In evaluating the neighbours, the function *ListCommodities* creates a list $\mathcal{P}_I$ of different commodities that flow between nodes i and j of an inefficient chain k, function *GenerateNeighbour* generates a neighbour of s' by applying the flow rerouting, and function *RemoveFirstCommodity* erases the first commodity of the list $\mathcal{P}_I$. Finally, the function *isItFeasible* returns "true" if a particular combination (k, i, j) leads to a feasible re-routing of commodity flow. The complexity of Algorithm 3 is $O(|\mathcal{N}|^2)$, where $|\mathcal{N}|$ is the number of nodes.

The sections below describe the components of the ILS in greater detail.

5.3.3. Neighbourhood topology-structures

The local search within ILS utilizes compound moves, new neighbourhood structures, and frequency based memory to escape from local optima by penalizing previously visited neighbours. For this purpose, an array $\vec{r}$ of size equal to the number of arcs is used to store the number of times an arc $(i, j) \in \mathcal{A}$ has participated in a local move. This number is used in the form of a penalty. Each time a better solution is found, $\vec{r}$ is re-initialized to zero (Paraskevopoulos *et al.*, 2012). The following formulae depicts the local move cost:

$$\Delta f_{move} = f(s') - f(s) + \rho \sum_{(ij) \in A} b_{ij} r_{ij}, \tag{27}$$

where ρ is a normalization parameter, i.e., the average cost of an arc obtained by the best solution found, and b_{ij} a binary parameter equal to 1 if arc (i, j) has participated in a local move, and 0 otherwise.

The cycle-based operator, originally proposed by Ghamlouche *et al.* (2003), considers each and every pair of nodes in the network and attempts to relocate the commodity flows on the links between the two nodes. The proposed neighbourhood search is applied on so-called "inefficient" chains, in systematic fashion, and evaluates all possible re-routings between two nodes in such

chains. Inefficient chains are constructed using the following formula:

$$In(ij) = \frac{\sum_{p \in P} x_{ij}^p c_{ij} + f_{ij} + g_i(x,c,z) + g_j(x,c,z)}{\sum_{p \in P} x_{ij}^p}, \tag{28}$$

where $In(ij)$ is an inefficiency measure of arc (i,j). The inefficient arcs are those whose $In(i,j)$ values are higher than the average arc-inefficiency of the current solution. Inefficient chains are formed by connecting such arcs. Each pair of nodes (and not only the adjacent) along these chains are the nodes that participate into the local moves.

5.3.4. Node upgrading decisions

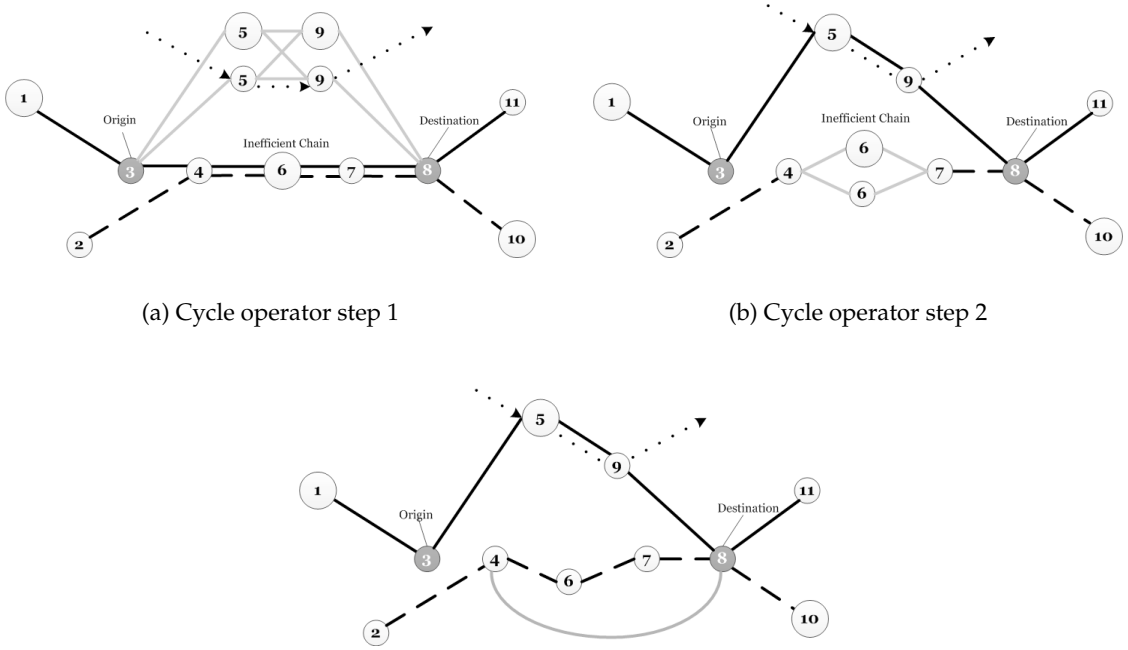
This section describes how node upgrading decisions are made within the ILS. Once an inefficient chain for flow rerouting is identified, all commodities that flow on the chain are selected. Alternative paths, that these commodities can be rerouted on, are then identified by the Dijkstra-like algorithm mentioned above. Within this algorithm, the labeling process is carried out such that each path considers upgraded and regular capacities of the nodes on the paths, and permanent labels are given to those that minimize the total cost. Hence the upgrading decisions on nodes are made as the alternative paths are iteratively "built". Once commodities have been rerouted from their original path to another, a further set of decisions are made for the upgraded nodes that appear on the original path as to whether downgrading results in better cost.

To be able to describe this process in greater detail, consider the example shown in Figure 8. In Figure 8a, two different commodities are considered, one flowing on the path shown by solid black lines (say ζ_1) and the other flowing on the path shown by the dashed lines (say ζ_2). Large nodes in the figure represent upgraded nodes, whereas the smaller ones are those which are not upgraded.

In this example, we assume that the path {3,4,6,7,8} define the part of the inefficient chain that will be subject to flow rerouting. Let an alternative path for rerouting ζ_1 be identified as {3,5,9,8}. Both the arcs and the nodes on the alternative path are then duplicated. As seen in Figure 8a, although nodes 5 and 9 can accommodate the "dotted" commodity flow, it may still be worth upgrading one (or both) of the nodes to reduce the total cost. Figure 8b shows the next iteration, assuming that only node 5 is to be upgraded. At this point, node 6 on the original path is checked to see whether it should be downgraded to further reduce costs, as the flow running through this node has now been reduced due to the rerouting. Finally, Figure 8c shows that the flow of commodity ζ_2 is rerouted on an alternative path shown by the grey lines between nodes 4 and 8. At this point, the link costs of the original path are penalized to prevent the procedure from rerouting the commodity to its current path.

5.3.5. Perturbation

To enable the search to escape from local optima, a perturbation procedure is applied whenever the local search has performed ψ iterations without an improvement in the objective function value. The perturbation strategy is to apply the local search introduced in Section 5.3.3 but with a different objective representing diversified solution structures. In particular, an array $\vec{h}$, similar to $\vec{r}$ introduced in Section 5.3.3, is used to store the number of times an arc has participated in a local move. The only difference between $\vec{r}$ and $\vec{h}$ is that the latter is not re-initialized every time a better local optimum is found, hence it retains a track record of the search history. To avoid large



(a) Cycle operator step 1

(b) Cycle operator step 2

(c) Cycle operator step 3

Figure 8: A typical cycle-based local search operator

numbers, we perform a scaling down in all the elements of array $\vec{h}$, as soon as one of the elements of the array reaches a large number Q. To guide the perturbation towards unexplored solutions in the search space, the cost of each arc (i, j) is multiplied by its score h_{ij} and the perturbation is applied by considering the revised cost of each arc. The perturbation procedure is applied for $\psi/2$ consecutive iterations, enough to be able to produce diversified solution structures but not long enough to "forget" the existing track record. The goal is to enable new arcs to participate to the current solution, which may have been neglected within the local search procedure, or even globally by the overall solution framework.

6. Computational Experiments

6.1. Generation of benchmark instances

To evaluate the proposed methods, new sets of cMCNDP benchmark instances were generated by extending the 43 original MCNDP instances described in Gendron and Crainic (1994, 1996). In particular, sets C and C+ were considered as "generating" instances, that include either 20, 25, 30 or 100 nodes, with the number of commodities ranging from 10 to 400, and the number of arcs from 100 to 700. The generating instances include information pertaining to the MCNDP, that is, the network structure (nodes and links), commodities (origins, destinations and amount), fixed design costs, and variable flow costs. Additional parameters required for the cMCNDP, in particular those related to nodes, were generated using the following ideas.

The main goal was to define the initial capacity, upgrading capacity, upgrading cost and the unit congestion cost, in a way that the upgrading decisions will be driven by cost savings, as opposed to accommodating any excessive flow that the initial capacity of a node may not cope with. The parameters generated are node-specific, to reflect the practical situations where some nodes may be more expensive to upgrade, busier than others, or larger or smaller in terms of capacity. The heuristic algorithm was used to define the node parameters. The main idea behind this process was to identify the maximum and minimum flows that a node can accommodate by inspecting several good-quality and diverse solutions produced by the heuristic and comprising set B, and calculating the respective flow costs that each node has to cope with. The objective function calibration parameters were set as follows: $D_i = 5$ and $F_i = 1$ for all $i \in \mathcal{N}$, and $\alpha = 3$, $\beta = 1$. The heuristic was run with $\psi = 30$, $\theta_{max} = 4$ and $\mu = 40$. For this experiment, the algorithm was not allowed to converge; instead it was run for three generations of the recombination process.

Initial capacities of the nodes are defined as 1.2 times maximum flow that flows into a node, as calculated from the elite pool of solutions B produced by the heuristic. The additional 20% is used as a safety margin to allow for higher values of flow, which might occur in solutions not discovered by the heuristic. As for incremental (upgraded) capacities, two settings were used: 30% of the initial capacity representing a minor upgrade investment, and 80% of the initial capacity corresponding to a major investment. For the general experiments, parameters α, β, and F_i remained at 3, 1 and 1 respectively. In an effort to normalize the order of magnitude of node and the arc costs, D_i was defined as the maximum unit cost of flow that a node i can accommodate. More specifically, if for a given solution s, y^s_{ij} is as defined before and $(x^p_{ij})^s$ is the amount of flow of commodity p on arc (i, j), then

$$D_i = \max_{s \in B} \left\{ \frac{\sum_{j \in \mathcal{N}_i^-} f_{ji} y^s_{ji} + \sum_{j \in \mathcal{N}_i^-} c^p_{ji} (x^p_{ji})^s}{\sum_{j \in \mathcal{N}_i^-} (x^p_{ji})^s} \right\}. \tag{29}$$

Finally, the upgrading cost is defined using the idea presented in Figure 9. In particular, this figure shows two graphs of the congestion function (14) for a particular node, one with and one without upgrading. The upgrading cost corresponds to an intersection point for these two functions, one that is between $[0.5c^0_i, c^0_i]$. We have chosen these values to be 0.6, 0.7 and 0.8 of the initial capacities, and calculated the upgrade cost as a result of solving a system of equations for calculating the intersection point for the three possible values.

Instance generation included therefore six problem instances for each of the original MCNDP instances. Each generating instance is further characterized by a node parameter shown by (Ω_1, Ω_2), where Ω_1 corresponds to the two upgrading capacities (either 30% and 80% of the initial node capacity), and Ω_2 reflects the three incremental cost settings for the two capacity options. This resulted in the generation of a total of $43 \times 6 = 258$ problem instances, which are available for download in the following online repository: `http://www.apollo.management.soton.ac.uk/cMCNDPlib.htm`. All experimentation has been performed by using CPLEX 12.1 running on single thread, on a computing cluster with 2.4 GHz and 64 GB RAM.

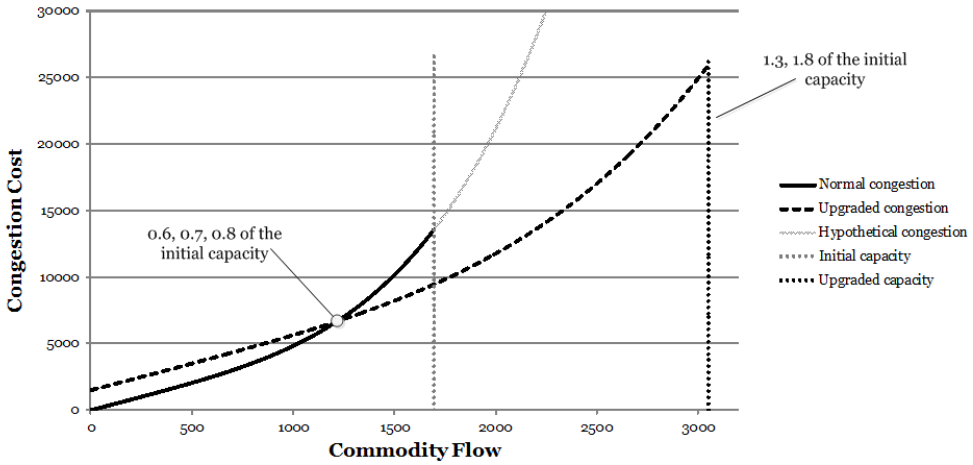


Figure 9: Congestion graphs with and without upgrading

6.2. Computational experimentation with the MISOCP reformulation

This section reports of our computational experience with solving $\mathcal{M}_{MISOCP}$, where all 258 instances were attempted to be solved using CPLEX. To be able to see the effect of the computational running time on the solution quality, two limits of 1000 and 3600 seconds are imposed on the solution time. Table 1 presents average results, for each generating instance, the optimality gaps for each set of MCNDP instances, averaged across the six cMCNDP instances contained within each set. Full results are available and can be found at `http://www.apollo.management.soton.ac.uk/cMCNDPlib.htm`.

As can be seen from Table 1, most instances can be solved to optimality using the MISOCP reformulation of the cMCNDP, shown by 0.00% or 0.01%, both of which are within the default optimality tolerance settings of CPLEX. Instance set c56 had one instance for which CPLEX was unable to derive an integer feasible solution within the computational time limit of 1000 seconds. A similar situation was observed for all instances in sets c62 and c64. When the time limit was increased to an hour, all instances were either solved to optimality or a feasible integer solution with a gap of at most 2.9% was identified.

6.3. Computational experimentation using the evolutionary algorithm

6.3.1. Parameter calibration

The proposed evolutionary algorithm uses five parameters; the number λ of initial solutions examined to produce the pool R of solutions, the size μ of the R, the size κ of the C, the maximum number ψ of local search iterations without an improvement in the solution quality, and the maximum number ϑ_{max} of CPLEX calls for which an improvement in the current solution is not observed. The termination criterion for the algorithm is a computational time limit of one hour. The scaling parameters γ and ρ are self-adjusted during the solution process, and are equal to the average cost of an arc in the current best solution found. The parameter λ was set equal to 500, as it was found not to have significant impact in the algorithm's performance. A further setting of $\kappa = 3$ is used since it should be relatively small to ensure that a large number of possible combinations of solutions, and larger than 2 to enhance the recombination process. The large number

Table 1: Average optimality gaps for each group of cMCNDP instances

Generating Instance	Average Optimality Gap %		Generating Instance	Average Optimality Gap %	
	1000 sec	3600 sec		1000 sec	3600 sec
25-100-10-V-L	0.00	0.00	100-400-10-F-L	3.70	1.42
25-100-10-F-L	0.00	0.00	100-400-10-F-T	3.50	2.74
25-100-10-F-T	0.00	0.00	100-400-30-V-T	0.21	0.13
25-100-30-V-T	0.01	0.01	100-400-30-F-L	2.10	1.07
25-100-30-F-L	0.01	0.01	100-400-30-F-T	3.72	2.90
25-100-30-F-T	0.01	0.01	c49	0.01	0.01
c33	0.00	0.00	c50	0.05	0.01
c35	0.01	0.01	c51	0.01	0.01
c36	0.01	0.01	c52	0.63	0.33
c41	0.01	0.01	c57	0.01	0.01
c42	0.01	0.01	c58	0.00	0.00
c43	0.01	0.01	c59	0.01	0.01
c44	0.01	0.01	c60	0.01	0.01
c37	0.66	0.24	c53	0.31	0.15
c38	2.63	1.66	c54	3.54	0.74
c39	0.19	0.03	c55	0.32	0.20
c40	1.79	1.22	c56	2.69*	0.83
c45	0.16	0.05	c61	0.60	0.36
c46	2.42	1.71	c62	N/A	2.43
c47	0.08	0.01	c63	0.44	0.30
c48	1.34	0.85	c64	N/A	1.57
100-400-10-V-L	0.01	0.01			

*: One instance is unsolved
N/A: All six instances are unsolved

$Q = 2000$, which is the maximum value that an element of the array h takes, before a scaling down of all the elements takes place.

Parameters ψ and ϑ_{max} should be defined so as to keep the total number of local search iterations balanced with the number of generations produced within the available computational time limit. Our preliminary experimentation has showed that values of ϑ_{max} equal to 6, 7, and 8 were appropriate. Table 2 shows the results of the parameter calibration experimentation conducted on C and C+ sets of benchmarks instances to determine the best values of ϑ_{max}, ψ and μ with respect to problem size. The instances are classified into six groups, according to the size of problems, and the label for each group, shown in the first line of Table 2, is a vector showing the number of nodes, the number of arcs and the number of commodities. For the experimentation, a single problem from each group was chosen, using the (0.3, 0.6) node configuration. As for the values tested, for large scale problems the reference set was assigned relatively small values and ψ was assigned high values, whereas the settings were the opposite for the small to medium scale instances.

During the experimentation, it has been observed that small to medium scale problem instances favoured a fewer number of local search iterations and a larger reference set, whereas large scale problem instances needed more local search iterations with a relatively small reference set to produce the best results. The reason for that is the fact that local search is not capable to show its strength in small scale problems, where the solution neighbourhoods are very small, and the evolutionary strategy seems to have a significant impact. The results given in Table 2 show that there is no significant variation of the solution costs derived from different parameter sets on the same problem instance problem, showing that the algorithm's performance is robust across different parameter settings. However, the best result for each different group was used as the fixed pa-

Table 2: Calibration of parameters ϑ_{max}, ψ and μ of the evolutionary algorithm

Group	**25-100-(10&30)**		**20-(230&300)-40**		**20-(230&300)-200**	
	$\vartheta_{max}, \psi, \mu$	25-100-30-FT	$\vartheta_{max}, \psi, \mu$	c36	$\vartheta_{max}, \psi, \mu$	c39
	6,50,40	210808.29	6,50,30	1489135.21	6,70,20	256166.62
	6,50,50	210835.89	6,50,40	1488987.80	6,70,30	258577.84
	6,50,80	210774.09	6,50,50	1486110.50	6,70,40	256486.94
Parameter	7,40,40	210964.53	7,40,30	1488493.88	7,60,20	256308.29
Sets	7,40,50	211879.33	7,40,40	1488722.68	7,60,30	258723.96
	7,40,80	210808.27	7,40,50	1487995.55	7,60,40	256823.14
	8,20,40	211065.17	8,30,30	1489397.69	8,40,20	257013.35
	8,20,50	210872.88	8,30,40	1488902.80	8,40,30	258389.38
	8,20,80	211375.76	8,30,50	1486607.47	8,40,40	256907.76
Best	6,50,80	210774.09	6,50,50	1486110.50	6,70,20	256166.62
Group	**100-400-(10&30)**		**30-(520&700)-100**		**30-(520&700)-400**	
	$\vartheta_{max}, \psi, \mu$	100-400-30-FT	$\vartheta_{max}, \psi, \mu$	c58	$\vartheta_{max}, \psi, \mu$	c64
	6,70,20	338884.30	6,70,20	183074.46	6,70,10	343397.28
	6,70,30	340756.68	6,70,30	183140.83	6,70,20	343259.43
	6,70,40	339872.22	6,70,40	183337.17	6,70,30	344032.84
Parameter	7,60,20	339654.95	7,60,20	182453.26	7,60,10	342604.96
Sets	7,60,30	339013.72	7,60,30	182812.77	7,60,20	344646.10
	7,60,40	340527.21	7,60,40	182597.53	7,60,30	344910.39
	8,40,20	339826.27	8,40,20	183058.25	8,50,10	344203.15
	8,40,30	338979.61	8,40,30	182597.53	8,50,20	343826.33
	8,40,40	340559.80	8,40,40	183632.89	8,50,30	345033.68
Best	6,70,20	338884.30	7,60,20	182453.26	7,60,10	342604.96

rameter setting for that particular group. This setting is applied to solve the rest of the problems in the group using a single run of the algorithm.

6.3.2. Computational results with the evolutionary algorithm

This section presents the results of the experiments where the evolutionary algorithm was compared with the results obtained using CPLEX. Full results are given in Table A.2 in the online appendix. A summary of the results is given in Table 3, showing averages for instances grouped into four on the basis of the number of nodes, being 20, 25, 30 and 100 shown under the first column. In this table, the column shown by "Avg. Total Cost" shows the average total cost of all the instances with the corresponding number of nodes, obtained within the time limits of 1000 and 3600 seconds. In this table, we also report the average number of nodes upgraded under column "UN". The column titled "Opt. Ratio" denotes the ratio of the number of instances solved to optimality out of the total number of instances tested within that group. The final column titled "Deviations (%)"shows the percentage deviations of the solution values obtained with the heuristic from those obtained by CPLEX.

In Table 3, we are not able to report any deviation values for the 30-node instance group running under a limit of 1000 seconds as CPLEX wasn't able to produce any solutions. The comparison results show that the heuristic was able to find solutions that are generally within 1%–2% of the optimal or near-optimal solutions identified by CPLEX. The average deviation across the instances shown is 1.12%. Out of the 43 instances shown in Table 3, the evolutionary algorithm was able to produce solutions with deviation below 1.00% for 29 instances. For the 25 instances for which optimal solutions were identified by CPLEX, the evolutionary heuristic was able to discover six of these.

Table 3: Summary of the comparison results of the evolutionary algorithm with CPLEX

Nodes	CPLEX				Evolutionary Algorithm				Deviations (%)	
	Avg. Total Cost				Avg. Total Cost					
	1000 sec.	3600 sec.	UN	Opt. Ratio	1000 sec.	3600 sec.	UN	Opt. Ratio	1000 sec.	3600 sec.
20	718089.6	718016.2	5.8	10/15	721457.6	720853.4	6.2	4/15	0.47	0.39
25	224316.2	224316.2	8.0	6/6	224509.7	224509.7	7.6	2/6	0.09	0.09
30	218804.2*	245114.0	8.9	7/16	250472.1	248272.1	10.6	0/16	-	1.27
100	269688.0	269301.2	13.8	2/6	272868.2	271802.2	14.6	0/6	1.17	0.92

*3 out of 16 solutions not found

6.4. Further numerical insight

6.4.1. Impact of congestion on the MCNDP solution

In this section, we provide further numerical insight on the impact that costs of congestion at nodes may have on the overall solution structure. The full results are presented in Table A.2 in the appendix, a summary of which are given in Table 4 where instances are grouped into four as in the previous section. These tables report statistics on the flow cost (AC), congestion cost (CC), total cost (TC) and the total number UN of upgraded nodes in the network. For a given instance, tests were conducted in the following way. First, the classical MCNDP is solved (columns 2–3 in Table 4). Then, using the solution of the MCNDP, a "manual" reconfiguration of the network to reduce the resulting congestion in the network by upgrading some nodes (columns 4–6) without changing the flows, which we call UpMCNDP. Finally, a cMCNDP is solved on the same instance (columns 7–10). The last two columns Dev1 and Dev2 show the percentage deviation of solution values of MCNDP and UpMCNDP from cMCNDP, respectively. The solutions for the MCNDP are taken from Paraskevopoulos *et al.* (2014).

Table 4: Summary of comparison results between the classical MCNDP and the cMCNDP on benchmark instances of Gendron and Crainic (1994, 1996)

Group	Avg. MCNDP		Avg. UpMCNDP			Avg. cMCNDP				Deviations (%)	
	AC	CC	CC	TC	UN	AC	CC	TC	UN	Dev1	Dev2
25	94613.0	151075.3	139760.9	234373.9	12.0	98045.6	126270.5	224316.2	8.0	−9.53	−4.48
20	291211.9	714609.3	604777.3	895989.2	14.7	339727.3	378285.6	718012.9	6.1	−40.08	−24.79
100	115502.7	189132.9	185185.3	300687.9	19.7	121178.0	148123.2	269301.2	13.0	−20.78	−11.65
30	94492.9	310417.8	240630.2	322717.5	24.3	119746.4	125367.6	245114.0	8.7	−65.19	−31.66

As Table 4 shows there are high congestion costs associated with the best solutions obtained for the classical MCNDP, which is obvious since congestion is not explicitly considered within this problem. The UpMCNDP approach is able to improve the situation by relieving congestion, implying improvements in the total cost from around 1% to 120%. The cMCNDP solution, on the other hand, is able to substantially improve on the MCNDP solutions, with the savings in cost ranging anywhere from 4.08% to beyond 200%. Interestingly, the flow cost (AC) in some cases, is seen to increase with cMCNDP but this is at the expense of obtaining a better TC. Finally, we observe that the number of upgraded nodes in cMCNDP is reduced in comparison to UpMCNDP, indicating that there is no need for upgrading so many nodes in the system. The results suggest that, instead of only upgrading nodes, a combination of upgrading with rerouting flows paths is a better way of alleviating congestion in the network.

To shed more light into the way in which the solutions themselves change under from the tradi-

Table 5: Statistics on capacity utilization of nodes for several problem instances

Instances	Average Capacity Utilization of Nodes							
	UpMCNDP				cMCNDP			
	Avg.	Max.	Min.	StDev.	Avg.	Max.	Min.	StDev.
20,230,40FT	0.530	0.715	0.229	0.206	0.484	0.641	0.245	0.157
20,300,40VL	0.591	0.845	0.260	0.181	0.504	0.641	0.183	0.159
30,520,100VT	0.591	0.965	0.404	0.101	0.473	0.653	0.274	0.130
100-400-10-VL	0.477	0.909	0.129	0.151	0.428	0.818	0.115	0.189
25-100-30-FT	0.567	0.813	0.459	0.083	0.539	0.632	0.265	0.148
25-100-10-FT	0.351	0.570	0.089	0.152	0.307	0.585	0.074	0.206

Table 6: Statistics on capacity utilization of arcs for several problem instances

Instances	Average Capacity Utilization of Arcs							
	UpMCNDP				cMCNDP			
	Avg.	Max.	Min.	StDev.	Avg.	Max.	Min.	StDev.
20,230,40FT	0.525	1.000	0.092	0.285	0.465	1.000	0.035	0.286
20,300,40VL	0.316	1.000	0.053	0.215	0.276	0.704	0.053	0.158
30,520,100VT	0.676	1.000	0.070	0.286	0.471	1.000	0.040	0.281
100-400-10-VL	0.544	1.000	0.056	0.311	0.519	1.000	0.014	0.344
25-100-30-FT	0.621	1.000	0.040	0.409	0.577	1.000	0.019	0.403
25-100-10-FT	0.624	1.000	0.074	0.396	0.628	1.000	0.049	0.410

tional MCNDP to the cMCNDP, we also present statistics on average (Avg.), minimum (Min.) and maximum (Max.) capacity utilization of nodes and arcs in Tables 5 and 6 for six different instances, as well as the standard deviations (StDev.). As this table shows, average capacity utilization on nodes decreases when the upgrading decisions at nodes are made and reaches lower levels in the cMCNDP solutions. Solving the cMCNDP typically results in an even distribution of the flows on the nodes and alleviates congestion, which consequently reduces arc capacity utilization as well.

6.4.2. Impact of different nodes' settings on the upgrading decisions

The reasoning behind defining node properties (as discussed in Section 6.1) was to investigate on the impact different nodes capacities and upgrading costs would have on upgrading decisions, and generally on the cMCNDP solution itself. To test this reasoning further, we provide results of some experiments in Table 7 based on a selected set of instances. The first column of Table 7 shows the generating MCNDP instances with varying node properties in the second column. The third column of the Table gives the status of the solution. In particular, "Optimal" or "Opt.Tol." indicates that an optimal solution was found. Otherwise the solutions are just "Feasible". The last two columns are as defined previously.

As seen from Table 7, as the upgrade costs increase, the number of upgraded nodes naturally decreases. In this case, it is cost-effective to reduce the investment in nodes at the expense of increased congestion. The reason behind the reduced number of upgraded nodes can be also attributed to the fact that when larger capacities at nodes are considered, more flows can be accommodated on a single node, removing the need to upgrade more nodes.

Table 7: Computational results on problem instances with different node properties

Generating Instance	Node properties	Status	TC	UN
25-100-10-V-L	(0.3, 0.6)	Optimal	30374.58	12
	(0.3, 0.7)	Optimal	30867.40	10
	(0.3, 0.8)	Opt.Tol.	31519.57	8
	(0.8, 0.6)	Opt.Tol.	29635.94	11
	(0.8, 0.7)	Opt.Tol.	30371.01	9
	(0.8, 0.8)	Optimal	31240.62	5
100-400-30-F-L	(0.3, 0.6)	Feasible	130773.15	17
	(0.3, 0.7)	Feasible	131503.84	11
	(0.3, 0.8)	Feasible	132323.93	6
	(0.8, 0.6)	Feasible	129795.04	15
	(0.8, 0.7)	Feasible	131127.82	11
	(0.8, 0.8)	Feasible	132603.66	5
c44	(0.3, 0.6)	Opt.Tol.	1545474.58	8
	(0.3, 0.7)	Opt.Tol.	1567386.50	5
	(0.3, 0.8)	Opt.Tol.	1586228.26	3
	(0.8, 0.6)	Opt.Tol.	1517143.54	7
	(0.8, 0.7)	Opt.Tol.	1548781.10	5
	(0.8, 0.8)	Opt.Tol.	1580478.01	4
c51	(0.3, 0.6)	Opt.Tol.	129889.36	12
	(0.3, 0.7)	Opt.Tol.	131467.24	3
	(0.3, 0.8)	Opt.Tol.	131669.81	0
	(0.8, 0.6)	Opt.Tol.	128522.12	13
	(0.8, 0.7)	Opt.Tol.	130983.52	2
	(0.8, 0.8)	Opt.Tol.	131403.71	1

6.5. Real-life case study

To validate the proposed approach in a practical setting, we have applied it on a real-life (service) network design problem. The problem arises within the Polcorridor study (Polcorridor, 2006), which stems from a large European research project looking at the development of new rail-based intermodal transport solutions on the area of Northern Europe, and has already been considered in the relevant literature (Bauer *et al.* 2009, Andersen *et al.* 2009, Andersen and Christiansen 2009). The problem consists of designing rail services to operate over three countries, namely Poland, Austria and the Czech Republic, with the main hubs being located in two ports in Poland and one in Vienna, between which goods are to be sent, for forwarding either to the northern or the southern European networks. The original network consists of 17 nodes, including 11 internal nodes in the three countries mentioned above, and the remaining external nodes that are beyond the boundaries of these countries and not included in the design problem. Figure 10 shows the map of the geographical region in which the problem is defined and the nodes that are part of the design. The reader is referred to Polcorridor (2006) and the above references for further details on the problem.

The original aim of the Polcorridor study was to design periodic rail services in an intermodal setting. In our application, we look at the problem from a different point of view, namely that of congestion at nodes, where goods are transferred from one service to another at hubs. This is particularly the case at cross-border nodes between Poland and the Czech Republic, as the train services in the two countries are not compatible and freight will need to be moved from one train to another. We treat the commodities as a set of one-off shipments, hence are not concerned with the periodicity of the services, although such requirements can be reflected in the approach we

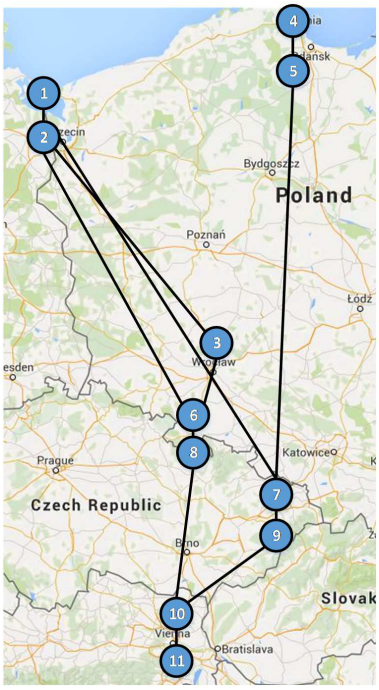


Figure 10: The Polcorridor supply system - Northern Europe

describe here. To apply the methodology, we first construct a time-space representation of the network, where each internal node $\{1, \ldots, 11\}$ is replicated for t periods. In our case t is equal to 150, to cover from the time at which the earliest commodity becomes available until the latest possible time when all commodities can be sent to their destinations. There are 40 different types commodities to be transported on the network. Each commodity originates from a node on the network in the set $\{1, \ldots, 11\}$ and becomes available at a given point in time, which corresponds to a unique node in the time-space network corresponding to the spatial and temporal attributes of the commodity. Under the assumption that the commodities will need to be transported as quickly as possible, a dummy node was introduced into the the time-space network for each geographical node $\{1, \ldots, 11\}$, and connected to the replicas of these nodes in each time period (see Figure 11 for a visual depiction). To enable the model to favour "earlier" rather than "later" arrivals to destinations, the arcs (i, j) that connect the replicated nodes and the dummy node have variable costs equal to $c_{i^t j_i^*} = (0.1t \sum_{i,j \in A} c_{ij})/|A|$, where i^t is the t^{th} copy of the internal node $i = 1, \ldots, 11$ in period t and j_i^* the dummy node of the same internal node i. Each arc has its own variable cost, which is equal to the distance between the two nodes it connects. We do not consider any fixed costs for this case study as the arcs that correspond to the rail tracks are already in place. The time-space network representation contains a total of 1671 nodes and 8385 arcs.

For defining the capacity of the each arc, the Evolutionary Algorithm was run in the same way as was done for designing the benchmark instances (see Section 6.1). More specifically, the maximum flow that an arc could accommodate throughout these runs was multiplied it by 1.5 plus 40, i.e., $v_{ij} = 1.5 maxFlow + 40$ was used as capacity of the arc (i, j). Following the same procedure discussed in Section 6.1, we defined the initial capacity, the incremental capacity, and the incremental cost for each node.

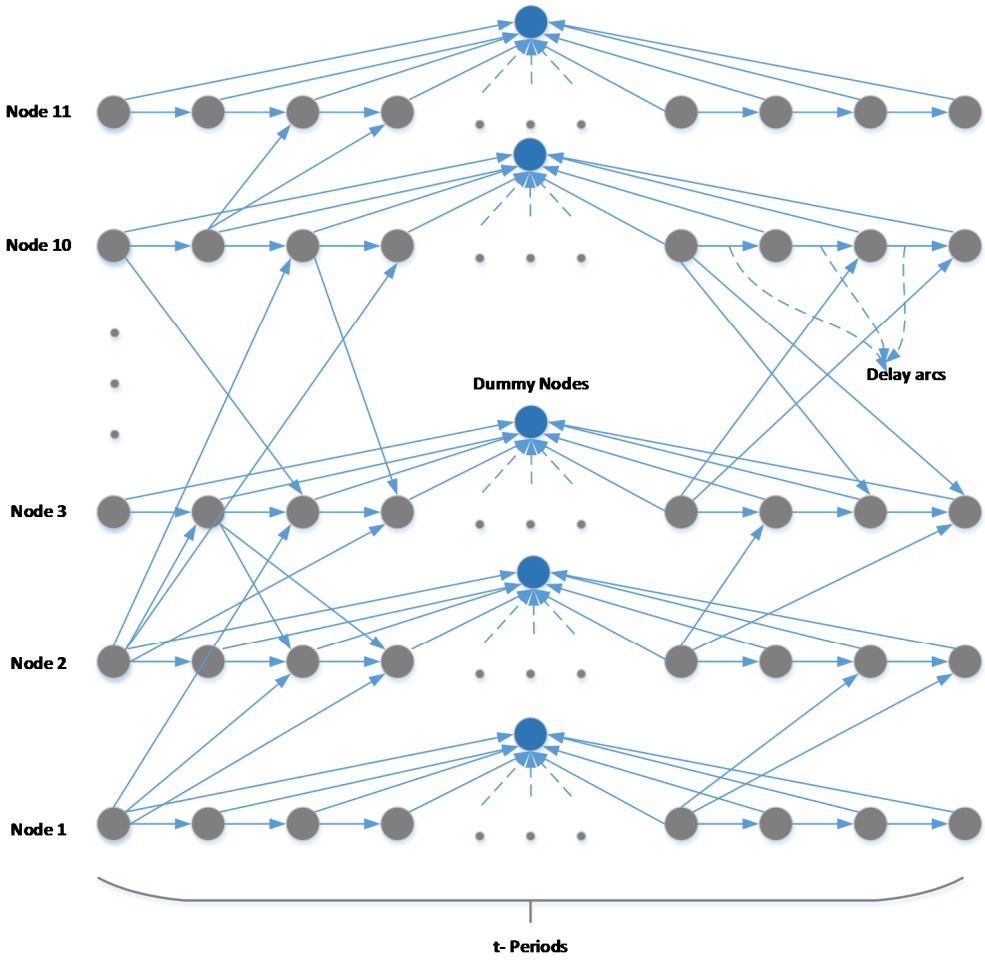


Figure 11: A typical Time-Space representation for the Polcorridor network

Table 8: Computational results on the Polcorridor case study (Google Maps)

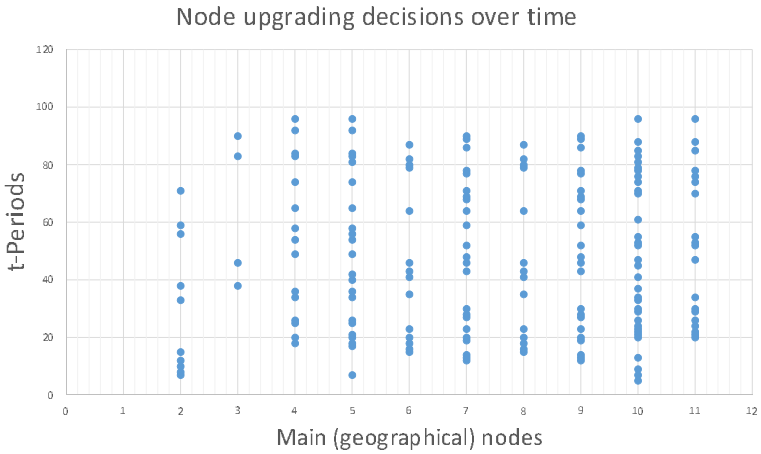


Figure 12: Upgrading decisions of nodes at different points in time

Table 8 reports the results of the Polcorridor case study. The table has an identical structure to that of Table 4. The reported results show that the congestion cost is significantly high in the MCNDP solution as compared to that of cMCNDP, i.e., the difference between the total cost of the solutions is -34.86%. Even when one uses the MCNDP solution and "manually" upgrades the congested nodes to reduce the resulting congestion in the network without changing the flows, the UpMCNDP solution deviates by -1.10% from to the solution obtained by solving the proposed cMCNDP.

The upgrading decisions (the number of which remained almost the same) are illustrated in detail in Figure 12. One can observe that most of the main (geographical) nodes need to be upgraded throughout a large part of the time horizon. Therefore rump up and rump down decisions can be made to minimize the total cost, following the upgrading decisions shown in Figure 12.

7. Conclusions

This paper presented and described solution methods for a variant of the multicommodity Network Design problem (MCNDP) where congestion at nodes is explicitly taken into account. The so-called congested MCNDP (cMCNDP) aims at finding flows for each type of commodity by activating suitable arcs and performing capacity upgrades at nodes, so as to minimize a function including operational and congestion costs, and by meeting capacity constraints on both arcs and nodes.

The paper also described a new set of benchmark cMCNDP instances that are publicly available and used for testing in this paper. Through extensive computational testing, the following observations can be made. Of the two solution approaches presented, the reformulation of the

problem is a viable way of solving small to medium (and a few large) scale instances to optimality. In particular, running the MISOCP reformulation on CPLEX, we were able to solve 133 of the 258 problem instances to optimality within an hour of computational time, including instances of up to 100 nodes, 700 arcs and 200 commodities. Our experience here shows that MISOCP is a good alternative to solve such problems, provided conic representations of the nonlinear terms are available.

Our results have also shown that, for the problem considered here, the efficient evolutionary algorithm that is equipped with long and short term memory structures, new local search operators and innovative solution recombination processes, yields satisfactory results. There are instances for which the evolutionary algorithm is able to find very good quality solutions in short computational times, which makes it a viable solution method for large scale cMCNDP instances for which MISOCP formulation might fail to perform well. We also stress that the flow subproblems solved within the evolutionary algorithm are nonlinear, and the resolution of these problems is made possible using the SOCP representations described here.

The experiments have also shown that, through solving the cMCNDP, a more even distribution of the flows can be obtained on the network, as well as reduction in congestion. Through reducing the maximum capacity utilization, the cMCNDP enables the network to accommodate more commodity flows without increasing congestion too much. This is particularly useful in the context of rail transportation, where one might be interested in adding more services to an existing service configuration. Similar results were obtained on a real-life case study arising in a service network design problem in Northern Europe, showing that a reduction of up to 35% of the total costs is possible through the proposed methodology.

The cMCNDP is relevant to transportation, telecommunications and production, where congestion issues are the main reason for deteriorating the efficiency of the operations as well as the service quality delivered to the customers. Further research on this problem can include defining toll pricing policies in motorways as a trade-off mechanism for alleviating congestion. On the algorithmic side, development of collaborative solution frameworks that combine both exact and heuristic components for solving the cMCNDP would be of interest.

Acknowledgments. Thanks are due to three anonymous reviewers for providing valuable comments on an earlier version of the paper. This research has been partially funded by the Engineering and Physical Research Council (EPSRC) and the University of Southampton Faculty Strategic Research Fund, both of which are gratefully acknowledged.

References

[1] Alizadeh, F. and Goldfarb, D. Second-order cone programming. *Mathematical Programming*, 95:3–51, 2003.

[2] Alvarez, A.M., Gonzalez-Velarde, J.L. and De-Alba, K. Scatter search for network design problem. *Annals of Operations Research*, 138:159–178, 2005.

[3] Andersen, J., and Christiansen M. Designing new European rail freight services. *Journal of the Operational Research Society*, 60:348–360, 2009.

[4] Andersen, J., Crainic TG and Christiansen M. Service network design with management and coordination of multiple fleets. *European Journal of Operational Research*, 193:377–389, 2009.

[5] Bauer, J., Bektaş, and Crainic TG. Minimizing greenhouse gas emissions in intermodal freight transport: an application to rail service design. *Journal of the Operational Research Society*, 61: 530–542, 2010.

[6] Bektaş, T., Chouman, M. and Crainic, T.G. Lagrangean-based decomposition algorithms for multicommodity network design problems with penalized constraints. *Networks*, 55:171–180, 2011.

[7] Bektaş, T., Crainic, T.G. and Morency, V. Improving the performance of rail yards through dynamic reassignment of empty cars. *Transportation Research Part C*, 17:259–273, 2009.

[8] Belotti, P., Malucelli, F. and Brunetta, L. Multicommodity network design with discrete node costs. *Networks*, 49:90–99, 2007.

[9] Ben-Ameur, W. and Ouorou, A. Mathematical models of the delay constrained routing problem. *Algorithmic Operations Research*, 1(2):94–103, 2006.

[10] Ben-Tal, A. and Nemirovski, A. *Lectures on Modern Convex Optimization: Analysis, Algorithms, and Engineering Applications*. SIAM, 2001.

[11] Chen, L. and Yang, H. Managing congestion and emissions in road networks with tolls and rebates. *Transportation Research Part B*, 46:933–948, 2012.

[12] Crainic, T.G. Service network design in freight transportation. *European Journal of Operational Research*, 122:272–288, 2000.

[13] Crainic, T.G. Long-Haul Freight Transportation. In R.W. Hall, editor, *Handbook of Transportation Science*, pages 451–516. Kluwer, 2nd edition, 2003.

[14] Crainic, T.G. and Bektaş, T. Dynamic empty car management in railyards. INFORMS Annual Meeting. Pittsburgh (PA), USA., 2006.

[15] Crainic, T.G. and Rousseau, J.M. Multicommodity, multimode freight transportation: a general modeling and algorithmic framework for the service network design problem. *Transportation Research B*, 20B:225–242, 1986.

[16] Crainic, T.G., Ferland, J.-A. and Rousseau, J.-M. A tactical planning model for rail freight transportation. *Transportation Science*, 18:165–184, 1984.

[17] Crainic, T.G., Li, Y. and Toulouse, M. A first multilevel cooperative algorithm for capacitated multicommodity network design. *Computers & Operations Research*, 33:2602–2622, 2006.

[18] Croxton, K.L., Gendron, B. and Magnanti, T.L. Variable Disaggregation in Network Flow Problems with Piecewise Linear Costs. *Operations Research*, 55:146–157, 2007.

[19] Demir E., Bektaş T., and Laporte G. A review of recent research on green road freight transportation. *European Journal of Operational Research*, 237(3):775 – 793, 2014.

[20] Elhedhli, S. and Hu, F.X. Hub-and-spoke network design with congestion. *Computers & Operations Research*, 32(6):1615–1632, 2005.

[21] Elhedhli, S. and Wu, H. A lagrangean heuristic for hub-and-spoke system design with capacity selection and congestion. *INFORMS Journal on Computing*, 22(2):282–296, 2010.

[22] Fan, L., Wilson, W.W. and Dahl, B. Congestion, port expansion and spatial competition for us container imports. *Transportation Research Part E,*, 48(6):1121–1136, 2012.

[23] Fan, L., Wilson, W.W., and Tolliver, D. Optimal network flows for containerized imports to the united states. *Transportation Research Part E,*, 46(5):735–749, 2010.

[24] Fernandez, J.E., De Cea, J. and Giesen, R. A strategic model of freight operations for rail transportation systems. *Transportation Planning and Technology*, 27:231–260, 2004.

[25] Fosgerau M. How a fast lane may replace a congestion tol. *Transportation Research Part B*, 45: 845–851, 2011.

[26] Franceschetti A., Honhon D., Woensel T.V., Bektaş T. and Laporte G. The time-dependent pollution-routing problem. *Transportation Research Part B*, 56:265–293, 2013.

[27] Frangioni, A. and Galli, L. and Scutella, M.G. Delay-constrained shortest paths: approximation algorithms and second-order cone models. *Journal of Optimization Theory and Applications*, 164(3):1051–1077, 2015.

[28] Gendron, B. and Crainic, T.G. Relaxations for Multicommodity Capacitated Network Design Problems. Technical Report CRT-965, Center for Research on Transportation, Montreal, 1994. Available at `http://www.iro.umontreal.ca/~gendron/Publications/CRT_965.pdf`.

[29] Gendron, B. and Crainic, T.G. Bounding procedures for multicommodity capacitated fixed charge network design problem. Technical Report CRT-96-06, Center for Research on Transportation, Montreal, 1996. Available at `http://www.iro.umontreal.ca/~gendron/Publications/CRT_9606.pdf`.

[30] Gerla, M. The design of store-and-forward networks for computer communications. *PhD Thesis, UCLA*, 1973.

[31] Ghamlouche, I., Crainic, T.G. and Gendreau, M. Cycle-based neighbourhoods for fixed-charge capacitated multicommodity network design. *Operations Research*, 51:655–667, 2003.

[32] Ghamlouche, I., Crainic, T.G. and Gendreau, M. Path relinking, cycle-based neighborhoods and capacitated network design. *Annals of Operations Research*, 131:109–134, 2004.

[33] Gürel, S. A conic quadratic formulation for a class of convex congestion functions in network flow problems. *European Journal of Operational Research*, 211(2):252–262, 2011.

[34] Hijazi, H., P. Bonami, G. Cornuejols and Ouorou A. Mixed-integer nonlinear programs featuring on/off constraints. *Computational Optimization and Applications*, 52:537–558, 2012.

[35] Holmberg, K. and Yuan, D. A Lagrangian heuristic based branch-and-bound approach for the capacitated network design problem. *Operations Research*, 48:461–481, 2000.

[36] Khaled, A.A., Jin, M., Clarke, D.B., and Hoque, M.A. Train design and routing optimization for evaluating criticality of freight railroad infrastructures. *Transportation Research Part B*, 71: 71–84, 2015.

[37] Koç C., Bektaş T., Jabali O. and Laporte G. The fleet size and mix pollution-routing problem. *Transportation Research Part B*, 70:239–254, 2014.

[38] Laval J.A., Cho H.W., Munoz J.C. and Yin Y. Real-time congestion pricing strategies for toll facilities. *Transportation Research Part B*, 71:19–31, 2015.

[39] Li, H., Song R., Jin, M., and He S. Optimization of railcar connection plan in a classification yard. Transportation Research Board 93rd Annual Meeting, Washington, D.C., 2014.

[40] Liu, J., Ahuja, R. K. and Sahin, G. Optimal network configuration and capacity expansion of railroads. *Journal of the Operational Research Society*, 59(7):911–920, 2008.

[41] Magnanti, T.L. and Wong, R.T. Network design and transportation planning: models and algorithms. *Transportation Science*, 1:1–55, 1986.

[42] Mahey, P., A. Benchakroun and Boyer F. Capacity and flow assignment of data networks by generalized benders decomposition. *Journal of Global Optimization,*, 20(2):169–189, 2001.

[43] Minoux, M. Network synthesis and optimum network design problems: models, solution methods and applications. *Networks*, 19:313–360, 1986.

[44] Miranda, G., Luna, H.P., de Camargo, R.S. and Pinto, L.R. Tree network design avoiding congestion. *Applied Mathematical Modelling*, 35:4175–4188, 2011.

[45] Ouorou, A., Mahey, P. and Vial, J.Ph. A survey of algorithms for convex multicommodity flow problems. *Management Science*, 46:126–147, 2000.

[46] Paraskevopoulos, D., Tarantilis, C.D. and Ioannou, G. Solving project scheduling problems with resource constraints via an event list-based evolutionary algorithm. *Expert Systems with Applications*, 39:3983–3994, 2012.

[47] Paraskevopoulos, D.C., Bektaş, T., Crainic, T.G. and Potts, C.N. A cycle-based evolutionary algorithm for the fixed charge capacitated multi-commodity network design problem. Technical Report CIRRELT-2014-36, CIRRELT, Montreal, 2014. Available at `http://www.cirrelt.ca/DocumentsTravail/CIRRELT-2014-36.pdf`.

[48] Petersen, E.R. Railyard modelling - Part I: Prediction of putthrough time. *Transportation Science*, 11:37–49, 1977a.

[49] Petersen, E.R. Railyard modelling - Part II: The effect of yard facilities on congestion. *Transportation Science*, 11:50–59, 1977b.

[50] Polcorridor. PolCorridor LOGCHAIN Project, Final Report. Institute of Transport Economics, Norway. 2006.

[51] Tirachini, A., Hensher, D.A. and Rose J.M. Multimodal pricing and optimal design of urban public transport: The interplay between traffic congestion and bus crowding. *Transportation Research Part B*, 61:33–54, 2014.

[52] Transportation Safety Board of Canada . Yard collision. Technical Report Toronto, Ontario., Railway Investigation Report R03T0047., 2003.

[53] Turnquist, M.A. and Daskin, M.S. Queueing models of classification and connection delay in railyards. *Transportation Science*, 16:207–230, 1982.

[54] Yao T, Friesz T.L., Wei M.M. and Yin Y. Congestion derivatives for a traffic bottleneck. *Transportation Research Part B*, 44:1149–1165, 2010.

Appendix: Supplementary Tables

Table A.1 shows computational results produced by the evolutionary heuristic in comparison to the solutions obtained by CPLEX. Where relevant, the designation **F** denotes if the fixed costs are more important relative to the variable costs, **V** if the variable costs are more important relative to the fixed costs, **L** if the arc capacities are loose or **T** if the arc capacities are tight. For instances for which CPLEX is unable to produce an optimal (or "Opt.Tol") solution within 1000 seconds, a second row is added to show the results obtained within 3600 seconds running time. In particular, the following abbreviations are used: Opt. for the optimum solutions, Opt.Tol. for the optimum solutions obtained within the default level of tolerance in CPLEX, Feas. for the feasible solutions and No sol. when none solution was obtained. These tables report, for each instance, and for CPLEX and for the evolutionary heuristic separately, the total cost of the best solution found (denoted TC), computational time (Time) in seconds, flow costs (AC), congestion costs (CC) and the total number of upgraded nodes (UN). The last column shows the percentage deviation of the solution found by the heuristic from that of CPLEX.

Table A1: Comparison results between the evolutionary algorithm and CPLEX

Instances	CPLEX						Evolutionary Algorithm					Dev
	Status	TC	Time (sec)	AC	CC	UN	TC	Time (sec)	AC	CC	UN	%
c33	Opt.Tol.	1017737.3	2	478868.9	538868.4	11	1017737.0	12	478868.9	538868.4	11	0.00
c35	Opt.Tol.	853084.8	7	422505.6	538857.4	5	853331.8	2215	431405.4	421926.3	5	0.03
c36	Opt.Tol.	1482124.2	11	693467.5	788656.7	4	1486110.0	985	685943.5	800167.0	5	0.27
c41	Opt.	1123342.1	2075	484290.6	639051.5	8	1123342.0	11	484290.6	639051.5	8	0.00
c42	Opt.Tol.	1369561.1	2996	699135.9	670425.3	9	1379365.0	384	742666.0	636698.8	8	0.71
c43	Opt.Tol.	1136210.9	3541	536876.5	599334.5	10	1136211.0	418	536876.5	599334.5	10	0.00
c44	Opt.Tol.	1545474.6	3494	723268.0	822207.6	8	1545475.0	401	723268.0	822206.6	8	0.00
c37	Feas.	256451.8	1001	121156.9	135294.9	3	258805.0	733	120799.3	138005.7	3	0.91
	Feas.	256355.6	3601	118497.1	137858.5	4	257875.2	3543	120875.1	137000.1	3	0.59
c38	Feas.	383448.8	1000	179673.7	203775.1	4	390643.0	987	189425.8	201217.7	4	1.84
	Feas.	383402.8	3600	178457.3	204945.5	5	388913.2	3432	177402.5	211510.7	5	1.42
c39	Opt.Tol.	254535.0	1001	120919.9	133615.1	2	256166.6	881	127678.8	128487.8	5	0.64
c40	Feas.	362745.0	1001	172441.0	190304.0	5	369742.4	996	177620.6	192121.8	6	1.89
	Feas.	362136.9	3600	167912.2	194224.7	5	368022.4	3589	175894.0	192128.3	7	1.60
c45	Opt.Tol.	195651.7	869	94448.7	101202.9	6	199082.9	738	94340.5	104742.4	5	1.72
c46	Feas.	308861.7	1001	145845.0	163016.7	4	314691.3	871	145575.6	169115.7	5	1.85
	Feas.	308656.3	3601	148443.0	160213.3	5	313771.5	2786	143300.8	170470.7	8	1.63
c47	Opt.Tol.	206366.5	864	98190.8	108175.7	5	209091.6	2976	100568.6	108522.9	3	1.30
c48	Feas.	275748.3	1001	130478.7	145269.6	6	282069.8	739	139812.4	142257.4	7	2.32
	Feas.	275507.2	3601	129411.1	146096.1	6	278305.3	2674	136072.3	142233.1	8	1.00
25,100,10VL	Opt.	30374.6	1	15229.2	15145.4	12	30443.9	1	15317.0	15126.9	12	0.23
25,100,10FL	Opt.	45154.8	1	19161.0	25993.8	2	45154.8	61	19161.0	25993.8	2	0.00
25,100,10FT	Opt.	83888.1	4	52893.9	30994.2	1	84036.0	157	53467.9	30568.1	1	0.18
25,100,30VT	Opt.Tol.	873455.0	2	370530.0	502924.9	10	873455.0	2	370530.0	502924.9	10	0.00
25,100,30FL	Opt.Tol.	102975.1	12	42320.0	60655.1	6	103194.5	556	42227.3	60967.2	5	0.21
25,100,30FT	Opt.Tol.	210049.5	4	88139.7	121909.9	17	210774.1	537	89943.8	120830.3	16	0.34
c49	Opt.Tol.	143270.6	23	70996.0	72274.6	10	144345.4	2422	72426.0	71919.4	12	0.74
c50	Opt.Tol.	235063.7	427	119528.0	115535.7	3	238690.7	2605	124401.9	114288.9	3	1.52
c51	Opt.Tol.	129889.4	146	60479.0	69410.4	12	131035.4	3310	62337.0	68698.4	13	0.87
c52	Feas.	237586.8	1001	118154.0	119432.8	8	242494.1	845	123403.9	119090.2	9	2.02
	Feas.	237545.0	3601	118583.0	118962.0	8	240954.3	2567	123226.9	117727.4	9	1.41
c57	Opt.Tol.	125694.8	50	64203.0	61491.8	5	126244.8	1234	67011.7	59233.1	4	0.44
c58	Opt.Tol.	181220.5	24	87342.0	93839.7	3	182453.3	3550	89396.0	93057.3	2	0.68
c59	Opt.Tol.	129729.5	202	62954.0	66775.5	8	130199.4	936	64968.0	65231.4	8	0.36
	Opt.Tol.	129729.5	202	62954.0	66775.5	8	129949.7	2451	64083.0	65866.8	8	0.17
c60	Opt.Tol.	144485.1	106	70661.0	73824.1	8	145020.0	3153	73514.0	71506.0	7	0.37
c53	Feas.	306543.8	1003	150563.9	155980.0	9	312009.9	889	152792.2	159217.7	11	1.75
	Feas.	306543.8	3602	150563.9	155979.9	9	308551.4	3598	156385.0	152166.5	14	0.65
c54	Feas.	406604.2	1004	184697.5	221906.6	3	407731.5	804	204189.4	203542.1	14	0.28
	Feas.	396769.2	3603	192000.1	204769.2	2	406544.3	3545	210937.1	195607.3	11	2.40
c55	Feas.	290383.3	1033	139456.2	150927.0	17	294844.2	763	143987.8	150856.4	18	1.51
	Feas.	290377.0	3632	139937.6	150439.5	16	292148.5	3245	146197.5	145951.0	21	0.60
c56	No sol.		1002				399355.1	875	203170.7	196184.3	10	n/a
	Feas.	388993.5	3606	188075.9	200917.6	14	396380.0	3324	200221.9	196158.1	11	1.86
c61	Feas.	269467.8	1003	128506.9	140960.9	8	275613.0	876	134764.7	140848.3	9	2.23
	Feas.	268622.2	3604	127801.9	140820.3	9	272872.4	3567	134531.6	138340.7	12	1.56
c62	No sol.		1002				378083.6	856	193545.1	194401.1	9	n/a
	Feas.	363242.9	3602	174608.5	188634.4	11	367236.5	3423	183349.0	183887.5	14	1.09
c63	Feas.	244515.5	1003	120739.3	123776.3	12	249494.3	910	124385.3	125109.0	12	2.00
	Feas.	244390.1	3604	121839.7	122550.3	11	247322.2	3452	123151.0	124171.2	13	1.19
c64	No sol.		1000				349938.9	878	175433.1	176311.1	11	n/a
	Feas.	335987.0	3608	166368.6	169618.4	10	342604.9	3433	175150.4	167454.6	11	1.93
100,400,10VL	Opt.Tol.	64154.3	8	30319.0	33835.3	13	64281.8	917	30289.9	33991.8	14	0.20
100,400,10FL	Feas.	51263.8	1023	29917.0	21346.8	17	53142.9	202	30714.0	22429.0	21	3.54
	Opt.Tol.	50783.6	3649	27643.0	23140.6	13	52149.6	2567	31151.0	20263.6	17	2.62
100,400,10FT	Feas.	117419.8	1001	66696.7	50723.1	0	123521.9	998	76084.8	47437.1	0	4.94
	Feas.	116972.4	3603	66884.8	50087.7	0	121501.9	3345	73276.9	48225.0	0	3.73
100,400,30VT	Feas.	916447.5	1001	397931.7	518515.8	19	923091.0	802	398319.1	524771.9	24	0.72
	Feas.	916345.4	3601	397590.2	518755.2	19	922711.4	3345	396083.7	526627.7	19	0.69
100,400,30FL	Feas.	130773.1	1025	57782.0	72991.1	17	131284.0	953	57654.8	73629.3	15	0.39
	Feas.	130773.1	3641	57782.0	72991.1	17	131284.0	953	57654.8	73629.3	15	0.39
100,400,30FT	Feas.	338069.6	1001	146598.8	191470.7	21	341887.5	965	153969.7	187917.8	13	1.12
	Feas.	336778.2	3605	146848.9	189929.2	16	338884.3	3067	157244.9	181639.4	23	0.62