Strengthening the integrality gap for the capacitated facility location problem with LP-based rounding algorithms

Pascal B.J. de Koster
4302508

The Netherlands, Delft, August 2016

Thesis for the Bachelor degree
Bachelor of Science in Applied Mathematics

Delft University of Technology
Delft Institute of Applied Mathematics,
Section of Optimization

Supervisor: Prof. Dr. Ir. K.I. Aardal
Thesis Committee: Prof. Dr. Ir. K.I. Aardal
Dr. D.C. Gijswijt
Dr. J.A.M. de Groot
Prof. Dr. Ir. A.W. Heemink

Abstract

This thesis studies the capacitated facility location problem, in which all clients have unit demand and all facilities have integral capacity. A linear relaxation is researched, with corresponding integrality gap bounded by a constant. Recently, such a linear relaxation has been found and proven using an LP-bounding algorithm. The formulation of the relaxation and the proof were very complex and intuitively hard to understand, however. Therefore, this thesis provides a simpler, more formulation and proof. This thesis has two main contributions. First, a structured overview of all the theory prior to the construction of the relaxation is provided. To do so, the minimum knapsack problem is treated, which is a simplified version of the capacitated facility location problem. An LP-based rounding algorithm is presented to illustrate general flow-network techniques for facility location problems. Second, the rounding algorithm for the capacitated facility location problem is illustrated and explained more accessible to readers less familiar with LP-based rounding algorithms. The existing rounding algorithm for the capacitated facility location problem is treated, illustrated and extended with *Matlab* code. The rounding algorithm proves an integral solution for the capacitated facility location can be constructed from the linear optimal solution, with cost no more than 288 times the cost of the fractional optimal solution. This proves that the integrality gap of the proposed relaxation is bounded by 288.

Contents

Nomenclature

Symbol	Description
CFL	capacitated facility location problem
c_{ij}	assignment cost between facility i and client j
$c(\mathbf{x}, \mathbf{y})$	total cost of solution $(\mathbf{x}, \mathbf{y})$
d_j	residual demand of client j
D	demand
$\mathcal{D}$	set of clients
$\mathcal{D}_H$	set of clients reachable in H by an unsaturated client
E	set of arcs
$\mathbf{f}$	flow feasible to $\mathbf{MFN}(\mathbf{g}, \mathbf{x}, \mathbf{y})$
G	complete bipartite graph
$h(i, j)$	amount of flow sent from client j sinked at facility i in $\mathbf{MFN}(\mathbf{g}, \mathbf{x}, \mathbf{y})$
$h(X, j)$	amount of flow sent from client j sinked at all facilities $i \in X$ in $\mathbf{MFN}(\mathbf{g}, \mathbf{x}, \mathbf{y})$
H	residual network
$\mathcal{F}$	set of facility possible locations
F	set of integrally opened facilities
$\mathbf{g}$	partial assignment
I	set of fully opened facilities
I_H	facilities reachable in H by an unsaturated client
IG	integrality gap
MKP	minimum knapsack problem
$\mathbf{MFN}(\mathbf{g}, \mathbf{x}, \mathbf{y})$	multi-commodity flow network
o_i	opening cost of facility i
$\mathcal{P}$	set of flow paths used by $\mathbf{MFN}(\mathbf{g}, \mathbf{x}, \mathbf{y})$
P	flow path in $\mathbf{MFN}(\mathbf{g}, \mathbf{x}, \mathbf{y})$
S	set of non-integrally opened facilities
u_i	capacity of facility i
u_i^A	capacity of facility i adapted in accordance with subset A
x_{ij}	assignment variable between facility i and client j
$(\mathbf{x}^*, \mathbf{y}^*)$	fractional solution
$(\hat{\mathbf{x}}, \hat{\mathbf{y}})$	semi-integral solution
$(\overline{\mathbf{x}}, \overline{\mathbf{y}})$	full integral solution
y_i	opening variable of facility i
$\mathbf{z}$	maximum fractional b-matching of G

1 Introduction

The capacitated facility location problem (CFL) is a well-investigated combinatorial optimization problem. The problem consists of a set of clients, which need to be supplied, and a set of possible locations to build facilities to supply the clients; for example, distribution centers. Opening a facility at a certain location has a corresponding opening cost. Supplying a client from a certain facility has a corresponding assignment cost, associated with the distance between the client and the facility. Furthermore, each opened facility can only supply a limited number of clients. In the studied version of the CFL, each client has a demand of 1, and each facility has an integral capacity, indicating the number of clients it can supply. The goal is to minimize the total cost such that all clients are supplied and no facility exceeds its capacity.

The CFL is an integer problem; facilities are either opened or closed. Partial opening is not allowed. Also each client can only be supplied by a single facility; it can not divide its demand over multiple facilities. Partly due to the integrality of the CFL, solving the problem is NP-hard. Instead of searching for an optimal solution, it often suffices to find a good solution, which approximates the cost of the optimal solution. To do so, algorithms based on local search have been constructed, which find an integral solution with cost at most a constant factor times the optimal cost [1, 2, 3]. The method described by Bansal et al. [3] constructed an integral solution with cost at most 5 times the optimal cost. All of these methods were based on local search algorithms, but no linear programming algorithm existed until 2014, when An, Singh and Svensson [4] presented a linear relaxation of the CFL with integrality gap of 288. Although this result may seem poor in comparison to the local search algorithms, the potential of linear programming relaxation for the CFL is still unexplored and may still improve in the future. Furthermore, linear programming relaxations can be solved or approximated very efficiently [4].

In the algorithm presented by An et al., the linear relaxation of the CFL is considered and strengthened with additional constraints. These constraints will prove redundant for any integral solution, but reduce the polygon of feasible solutions significantly. Currently, the constraints added by An et al. are abstract and the overall algorithm for constructing the integral solution is hard to understand. The proof for the bound of 288 is hard to follow and often lacks intuition. A more insightful and accessible explanation of the rounding algorithm is required, such that it may benefit a broader public.

This is the main incentive of this thesis. A more insightful explanation is presented for understanding the algorithm of An et al. is contributed. Also, the thesis elaborates on possible methods to define the additional, abstract constraints more concretely. A *Matlab* code is also provided to run the algorithm of An et al. in order to clarify the process and to show the algorithm can also be implemented.

In Chapter 2, the capacitated facility location problem is explained and illustrated, in Chapter 3, the minimum knapsack problem (MKP) is considered. This is a relaxed version of the capacitated facility location problem, which only takes the cost of the facilities into account and leaves out the connection costs between client and facility. A linear relaxation of the MKP is shown such that its integrality gap is bounded by a constant [5]. The method used for the minimum knapsack problem shows a useful approach for strengthening the linear relaxation and converting the optimal linear solution to an integral solution with bounded cost. Finally, in Chapter 5, the algorithm of An et al. is explained. The linear relaxation of the CFL is strengthened and it is proven that the strengthened formulation of the CFL has an integrality gap of at most 288.

2 Capacitated Facility Location

In this Chapter the investigated capacitated facility location problem is be defined. Also the linear relaxation of the problem is considered. Furthermore, the integrality gap is investigated without additional constraints. Finally a simple strengthening of the relaxed CFL is shown.

As discussed in the introduction, the capacitated facility location problem is an NP-hard integer optimization problem. In the problem, there are m clients and n possible locations for opening a facility. The set of clients will be called $\mathcal{D}$ and the set of facilities $\mathcal{F}$. Each client has one unit of demand, and each facility has a capacity of $U_i, i \in \mathcal{F}$ when opened. Each facility is associated with an opening cost o_i, $i \in \mathcal{F}$. Connecting a client j to a facility i comes with a cost of c_{ij}, $i \in \mathcal{F}, j \in \mathcal{D}$. Clients can only be connected to an opened facility. The connection cost can often be related to the distance between client and facility. The problem is to choose which facilities to open and which open facility each client should be assigned to. An example of the problem is illustrated in Figure 1.

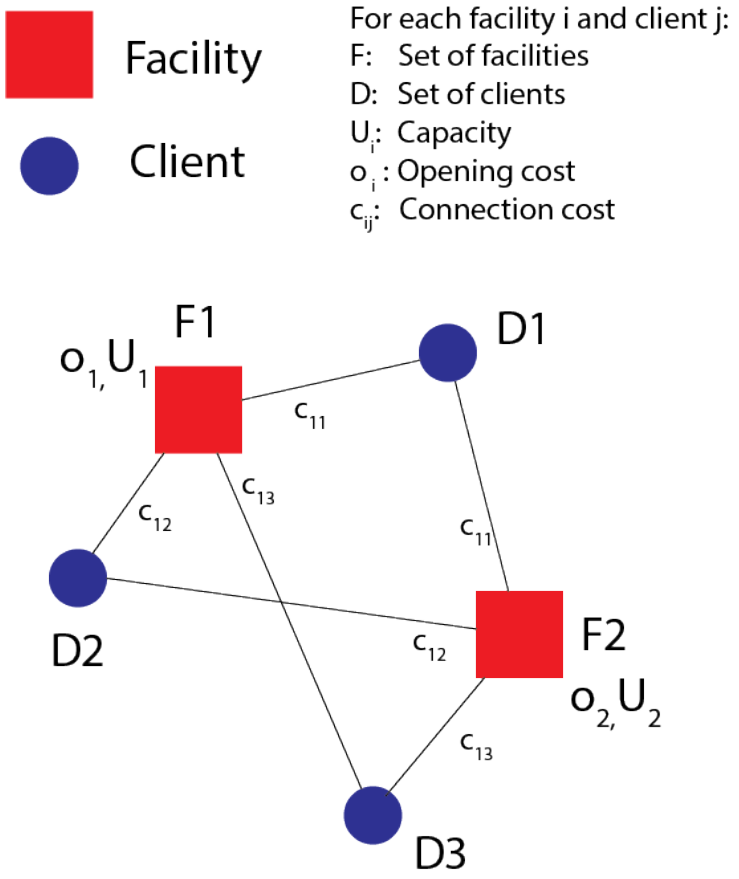


Figure 1: Illustration of the CFL. In this example, there are 3 clients and 2 facilities. The lines between facilities and clients indicate possible connections. Also, it must be decided what facilities to open.

In the studied version of the CFL, it is assumed that the triangle inequality holds for the connection cost c_{ij}. In its ordinary form, the triangle inequality states that the distance between two points a and c is always smaller when going directly than when first passing by an arbitrary point b.

$$d(a, c) \leq d(a, b) + d(b, c) \tag{1}$$

In the triangle inequality, d is an arbitrary metric. If the triangle inequality holds for the CFL, the connection cost between a client i and a facility j is smaller than the connection cost of any other route connecting client i and facility j through other clients and facilities. An example is shown in Figure (1).

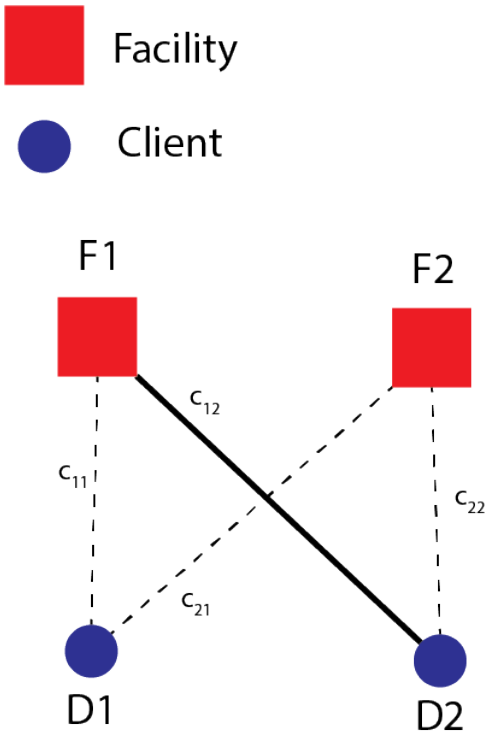


Figure 2: Illustration of the triangle inequality for the CFL. The direct path between F1 and D2 (marked solid) with cost c_{12} must be lower than the dotted path, which has cost $c_{11} + c_{21} + c_{22}$.

In many practical cases, the triangle inequality is a reasonable assumption, as the connection cost is often related to the distance. The triangle inequality will prove essential for the approximation algorithm based on linear programming of An et al., in Chapter 5.

Having defined the CFL and according notation, the integer linear programming (ILP) formulation can be defined. Any possible solution to the CFL can be expressed as an opening vector $\mathbf{y}$ and a assignment matrix $\mathbf{x}$.

$$y_i = \begin{cases} 1 & \text{if facility } i \text{ is open} \\ 0 & \text{if facility } i \text{ is closed} \end{cases} \tag{2}$$

$$x_{ij} = \begin{cases} 1 & \text{if client } j \text{ is connected to facility } i \\ 0 & \text{if client } j \text{ is not connected to facility } i \end{cases} \tag{3}$$

It is possible to express each constraint in terms of $\mathbf{x}$ and $\mathbf{y}$. The ILP formulation of the CFL is shown in Equations (4)–(7).

The objective function in Equation (4) can be expressed as the cost of used connections plus the cost of the opening the corresponding facilities. If a connection or a facility is not used, then respectively $x_{ij} = 0$ or $y_i = 0$. As shown in Equation (4) then the corresponding connection or opening cost gives no contribution. When i and j are connected, then $x_{ij} = 1$, so c_{ij} is contributing to the total cost and the same holds for the opening of each facility. The

$$\text{minimize} \quad \sum_{i\in\mathcal{F},j\in\mathcal{D}} c_{ij}x_{ij} + \sum_{i\in I} o_i y_i \tag{4}$$

$$\text{subject to:} \sum_{j\in\mathcal{D}} x_{ij} \leq U_i y_i, \qquad \forall i \in \mathcal{F} \tag{5}$$

$$\sum_{i\in\mathcal{F}} x_{ij} = 1, \qquad \forall j \in \mathcal{D} \tag{6}$$

$$y_i, x_{ij} \in \{0,1\}, \qquad \forall i \in \mathcal{F}, \forall j \in \mathcal{D} \tag{7}$$

Figure 3: ILP formulation of the capacitated facility location problem.

constraint in Equation (5) states that each facility i can supply no more clients than its capacity if it is opened ($y_i = 1$), and no clients when closed ($y_i = 0$). Equation (6) states that each client gets its unit of demand supplied from the set of facilities. Precisely one of the facilities should fully supply client j, $x_{ij} = 1$, and the other facilities should not be connected to the client, $x_{ij} = 0$. Finally, only integral opening and integral connections are allowed, which is stated by Equation (7).

2.1 Solving and approximating algorithms

Solving the CFL is NP-hard. As of yet, no algorithm exists to find an optimal solution in polynomial time [6]. Solving the capacitated facility location problem can be done in exponential time, however, by checking every possible solution. Each possible solution $(\mathbf{x}, \mathbf{y})$ can be brought forth by the matrix $\mathbf{x}$ alone: each client must be connected to a facility, open a facility if at least one client is connected to it. Opening any more facilities will only raise the total cost and is therefore never optimal. The optimal solution can be found by checking for all possible assignment matrices $\mathbf{x}$. Each of the $|\mathcal{D}|$ clients can be connected to $|\mathcal{F}|$ facilities. Therefore, $|\mathcal{F}|^{|\mathcal{D}|}$ possible solution must be checked on feasibility and cost. This will guaranteed give an optimal solution, if any exists, but the algorithm is very time consuming. For a set of 10 clients and 10 facilities, there are 10^{10} possible solutions to check. The number of possibilities grows exponentially with the number of clients. Due to its runtime this algorithm is very unpractical for all but the smallest instances of the CFL. In practice a branch-and-bound algorithm could be used. This algorithm is in general much faster, but the worst case run time is still exponential.

Instead of solving the CFL optimally, algorithms have been researched to find a good solution to the problem. These algorithms do not necessarily find an optimal solution, but construct a feasible solution approximating the optimal cost. Many of these algorithms are based on local search techniques. These techniques start at a feasible solution and each step move to a nearby feasible solution with lower cost. In the algorithm used by Aggarwal et al. [2], given a feasible solution, three possible operation were considered: closing a facility, opening a facility and both at the same time. After closing or opening facilities, the clients were reconnected to the opened facilities such that the connection cost was minimal. Consider the example shown in Figure 4. Given a problem with 2 possible facility locations and 2 clients, a feasible solution is given by connecting client 2 to facility 2 and client 1 to facility 1. The algorithm used by Aggarwal et al. has two nearby solutions: shut down facility 2 or shut down facility 1. If facility 2 is shut down, the clients are reconnected to the remaining facilities such that the cost is minimal. In this case, client 2 is connected to facility 1 (if the capacity of the remaining facilities allows it, otherwise, the nearby solution is infeasible) and the new connection is indicated by the dotted

line.

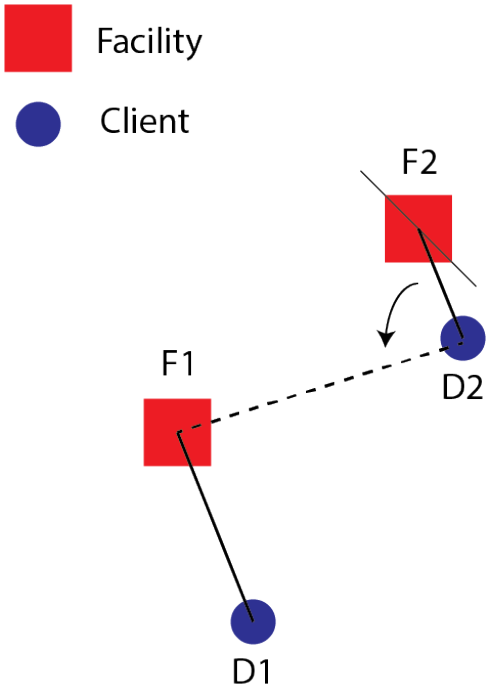


Figure 4: Illustration of a local search algorithm. Facility 2 is shut down. As consequence, client 2 is reconnected to facility 1.

Other local search algorithms may use different nearby solutions, but the general idea is the same as in the example. To ensure a polynomial runtime, it can be demanded that a nearby feasible solution is only accepted if it has cost at least ε better than the last cost, with $\varepsilon > 0$ an constant depending on the desired accuracy and speed of the algorithm. As mentioned in the introduction, the local search algorithms of Bansal et al. [3] was proved to find a solution with cost at most 5 times the optimal cost.

A completely different approximation method is based on the linear relaxation of the CFL. This thesis focuses on this type of method. The next section will elaborate on the basics of linear relaxation methods.

2.2 Linear relaxation of the CFL

Instead of using local search algorithms, the integer solution of the CFL can be approximated using the linear relaxation of the problem. For the linear relaxation, the integrality constraint in Equation (7) is relaxed to a linear one. The constraint, $y_i, x_{ij} \in \{0, 1\}$, is replaced by:

$$y_i, x_{ij} \in [0, 1], \quad \forall i \in \mathcal{F}, \forall j \in \mathcal{D} \tag{8}$$

Instead of demanding $\mathbf{x}$ and $\mathbf{y}$ to be integral, any value between 0 and 1 is allowed. Because all the constraints for the CFL are now linear, the relaxed CFL problem can be solved by means of linear programming (LP). The huge advantage of LP, is that these can be solved in polynomial ime. In practice one would often use a simplex-type method, which is not polynomial but efficient in practice. In the *Matlab* script in Appendix B, it is shown how the relaxation can be solved optimally using the simplex method. The idea about linear programming is to construct a space of feasible solutions. This space is constructed from the constraints, and is a polyhedron.

The simplex method steps from vertex to vertex of the polyhedron, in each step maintaining or lowering the cost of the feasible solution, until an optimal solution is found. Each feasible solution $(\mathbf{x}, \mathbf{y})$ is represented as a single vector. Note that the assignment $\mathbf{x}$ is a $n \times m$ (facilities times client) matrix and the opening vector $\mathbf{y}$ is a vector of length n (facilities). These can be written in a single vector as follows:

$$\left.\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \quad \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix}\right\} \rightarrow [\mathbf{yx}] = \begin{bmatrix} y_1 \\ \vdots \\ y_n \\ x_{11} \\ x_{21} \\ \vdots \\ x_{n1} \\ x_{12} \\ x_{22} \\ \vdots \\ x_{n2} \\ \vdots \\ x_{1m} \\ x_{2m} \\ \vdots \\ x_{nm} \end{bmatrix} \tag{9}$$

First the vector $\mathbf{y}$ is taken for the solution vector, then each of the columns of the matrix $\mathbf{x}$ is appended to the solution vector. In Equation (9), colors mark where each column ends up in the solution vector, which will be called $[\mathbf{yx}]$. An identical process happens for the cost vector $\mathbf{c}$, which is constructed from the opening cost o_i for opening a facility and c_{ij} for assigning a client to a facility. The cost vector $\mathbf{c}$ for the linear program is thus identical to $[\mathbf{yx}]$, when y is replaced by o and x by c. Note that then $\mathbf{c^T} \cdot [\mathbf{yx}]$ is indeed the same as $\sum_{i \in \mathcal{F}, j \in \mathcal{D}} c_{ij} x_{ij} + \sum_{i \in I} o_i y_i$.

Then all the constraints can be expressed in this solution vector. A linear program required matrices $\mathbf{A}$, $\mathbf{A}_{eq}$ and vectors $\mathbf{b}$, $\mathbf{b}_{eq}$. Here eq stands for 'equal', which becomes clear from the LP-formulation used for the *Matlab* script. The formulation is shown below and the corresponding *Matlab* script can be found on the first pages of Appendix B.

$$\begin{aligned} \underset{[\mathbf{yx}]}{\text{Minimize:}} \quad & \mathbf{c}^T \cdot [\mathbf{yx}] && (10) \\ \text{Subject to:} \quad & \mathbf{A} \cdot [\mathbf{yx}] \leq \mathbf{b} && (11) \\ & \mathbf{A}_{eq} \cdot [\mathbf{yx}] = \mathbf{b}_{eq} && (12) \\ & \mathbf{0} \leq [\mathbf{yx}] \leq \mathbf{1} && (13) \end{aligned}$$

Figure 5: LP formulation for computationally solving the linear relaxation of CFL.

Each row in the matrices $\mathbf{A}$ and $\mathbf{A}_{eq}$ with corresponding number in $\mathbf{b}$ and $\mathbf{b}_{eq}$ represents a constraint. Each constraint can be expressed in either the equality or the inequality matrix. For example, consider the constraint $\sum_{j \in \mathcal{D}} x_{1j} \leq U_1 \cdot y_1$. This can be expressed as follows. First the right hand side is moved to the left, $\sum_{j \in \mathcal{D}} x_{1j} - U_1 \cdot y_1 \leq 0$. Then this can be written as:

$$(-U_1) \cdot y_1 + 0 \cdot y_2 + \cdots + 0 \cdot y_n + 1 \cdot x_{11} + 0 \cdot x_{21} + \cdots + 1 \cdot x_{12} + \cdots + 1 \cdot x_{1m} + \cdots + 0 \cdot x_{nm} \leq 0$$

This can be rewritten as a vector multiplication:

$$\begin{matrix}[-U_1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 1 & \cdots & 1 & \cdots & 0] \\ [y_1 & y_2 & \cdots & y_n & x_{11} & x_{21} & \cdots & x_{12} & \cdots & x_{1m} & \cdots & x_{nm}]^T\end{matrix} \cdot \quad \leq 0$$

Note that the lower vector is equal to [**yx**]. The upper vector is row of matrix **A**, and this row has corresponding element 0 in vector **b**. The process for solving the linear relaxation of the CFL problem can be found in the first part of the *Matlab* script in Appendix B.

2.2.1 Integrality gap

The optimal linear solution of the linear problem is often fractional, and therefore not feasible for the integer CFL, however. Any solution feasible for the standard CFL on the other hand, is also feasible for the linear CFL, as the linearized version is a relaxation of the original problem (the integrality constraint is relaxed). Therefore, the optimal cost of the linear CFL is a lower bound for the optimal cost for the integral CFL.

Ideally, the fractional, optimal solution and the integral, optimal solution are close to each other. This is often expressed as the integrality gap (IG), i.e. the ratio between the integral and fractional, optimal solution:

$$IG = \frac{C_{int,opt}}{C_{frac,opt}} \tag{14}$$

In Equation (14), $C_{int,opt}$ denotes the optimal integral cost and $C_{frac,opt}$ denotes the optimal fractional cost. Note that the cost of the fractional solution is always a lower bound for the cost of the integral solution, therefore the integrality gap is at least 1. For actually calculating the integrality gap of a specific instance of the CFL, both the cost of the integral and fractional optimal solution are required. However, the exact value of the integrality gap for a certain instance is of minor importance. More importantly, it is desirable that the integrality gap for every instance of the CFL is bounded by some constant M. This would ensure the optimal integral solution is no worse than this constant times the cost of the fractional solution: $IG \leq M \Rightarrow C_{int,opt} \leq M \cdot C_{frac,opt}$.

It has been shown that the linear relaxation of the CFL in Equation (8) can be arbitrarily bad [5, 1]. The integrality gap of the CFL defined as in Equations (4)-(7) is not bounded by any constant. One of the issues causing this, is that an entire facility may have to be opened for a single client. For the integral solution, this facility is entirely opened, whereas for the linear solution, from Equation (6), $\sum_{j \in \mathcal{D}} x_{ij} \leq U_i y_i$, it suffices for this client if the facility is opened for $y_i = 1/U_i$. This also only brings but $1/U_i$ times the opening cost, whereas the integral solution has to pay for the full opening. The current linear relaxation is not strong enough to approximate the integral solution with a constant bound for the integrality gap. A possible solution lies in the strengthening of the linear relaxation.

2.2.2 Strengthening the linear relaxation

Strengthening of a linear relaxation can be done by adding additional constraints, such that the formulation better approximates the integral case. It is important that the extra constraint do not exclude feasible integral solutions, but only do so for fractional solutions. In the last section one of the issues causing a poor integrality gap was shown. Some facilities are opened for a small fraction to supply only a small number of clients. It is sometimes possible to prevent such small opening by adding an additional constraint. Consider the extra constraint in Equation (15).

$$x_{ij} \leq y_i, \quad \forall i \in \mathcal{F}, j \in \mathcal{D} \tag{15}$$

This constraint states that if a client j gets fractionally supplied by a facility i, then the facility should be opened for at least that fraction. If a client gets fully supplied by a single facility, the facility should be fully opened as well. Note that this constraint is satisfied by all integral solutions, because the constraint of Equation (5), $\sum_{j \in \mathcal{D}} x_{ij} \leq U_i y_i$, already demanded that y_i had to be greater than zero if some client was connected to it. Because y_i was integral and greater than zero, it had to be one.

This is just a single example of adding constraints to strengthen the linear relaxation. These additional constraints are still by far not enough to guarantee a good bound for the integrality gap, though. For example, if there are m facilities, then each can in theory be opened for $1/m$ part and each client can get $1/m$ part of its demand from each facility. Each client would then get its demand of $1 = m \cdot \frac{1}{m}$ satisfied. If all facilities but one were very cheap to open, and the last facility is extremely expensive, the expensive facility would only have to be opened for $1/m$ part, whereas in the integral solution, it may have to be opened fully in order to have enough total capacity. This strengthened formulation of CFL can still have a poor integrality gap [4].

It has proven quite hard to strengthen the CFL such that its integrality gap is bounded by a constant for all possible instances. No formulation was known, until 2014. In this year, An et al. [4] did succeed to strengthen the standard linear relaxation, and proved that their strengthened formulation had an integrality gap bounded by 288 for all instances. In their proof they showed an LP-based rounding algorithm to construct an integral solution with cost at most 288 times the cost of the optimal fractional solution. Because the optimal fractional solution is a lower bound for the optimal integral solution, the constructed integral solution is also at most 288 times the cost of the optimal integral solution. This is illustrated in Figure 6.

The strengthening of the linear relaxation and the LP-based rounding algorithm to find a feasible integral solution are respectively described in Chapters 4 and 5. First, a simpler variant of the CFL is researched. The minimum knapsack problem (MKP) is identical to the CFL when the connection cost between each client and facility are set to zero. First this problem will be relaxed and strengthened. Then using an algorithm, an integral solution is constructed and it is proven that the integrality gap of the strengthened knapsack problem is bounded by a constant. This problem is treated first as many rounding techniques used in the knapsack problem are also used in the algorithm of An et al. for the CFL problem. Furthermore, the algorithms for the MKP are intuitively better to understand.

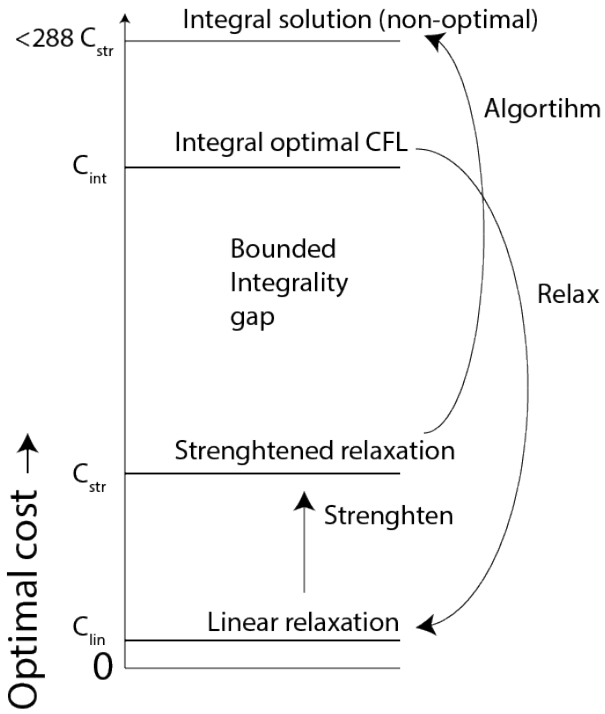


Figure 6: The optimal integral cost, C_{int}, for the CFL is unknown. To approximate it, the integral CFL is relaxed and strengthened. The strengthened relaxation has an optimal solution with lower cost than the integral optimal solution, but higher than the unstrengthened linear solution. The algorithm of An et al. produces an integral solution feasible to the original CFL, with cost at most 288 times the cost strengthened linear optimum. Therefore, $C_{int} \leq 288 \cdot C_{str}$.

3 Minimum Knapsack Problem

In order to better understand algorithms for strengthening the capacitated facility location problem, this section shows a strengthening of a much simpler problem related to the CFL: the minimum knapsack problem (MKP). First this problem will be introduced and solved linearly. Second, the linear problem is strengthened with additional conditions. Third, it is shown that the integrality gap of the strengthened problem is 2 and an algorithm is used to convert the fractional optimal solution from the strengthened, linear problem to an integer solution with cost no more than twice the fractional solution, and hence no more than twice the optimal, integer solution.

3.1 Introduction to the Knapsack Problem

The original knapsack problem consists of a set of items, where each item has its own value and weight, and a knapsack with a weight limit. The goal is to select a subset of items such that its total value is maximized without exceeding the weight limit of the knapsack. An item may not be chosen more than once. An example is shown in Figure 7.

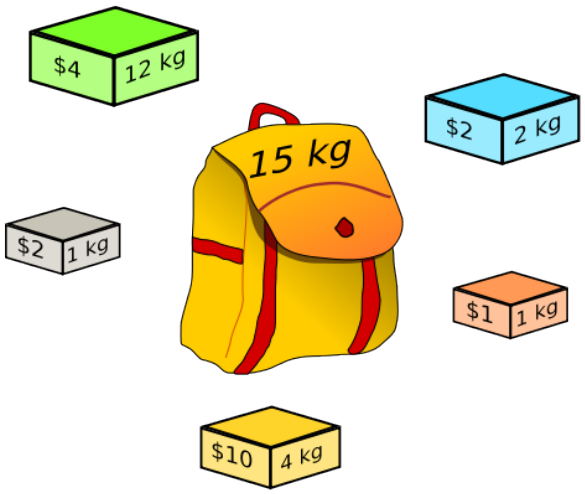


Figure 7: Knapsack problem. The goal is to choose a set of items with maximum value without exceeding the maximum weight limit. The solution of this problem includes all items except for the 12 kg item. The total weight is then 8 kg, which is less than or equal to the capacity of 15 kg, so the solution is allowed. The value of this optimal solution is $15.

The minimum knapsack problem investigated in this thesis is the reverse problem of the original knapsack problem. In the reverse problem, instead of maximizing the value without exceeding a weight limit, the goal is to minimize the weight given that the knapsack should contain at least a certain amount of value. For example, in the knapsack problem in Figure 7, a value of $4 may be demanded, which items should be chosen to minimize the total weight? For this example the solution consists of both $2 items for a total weight of 3 kg. Note that in this minimum knapsack problem, the knapsack has no maximum weight limit. The only restriction for this problem is the minimum value the knapsack should contain.

The minimum knapsack problem is closely related to the capacitated facility location problem. The minimum value demanded in the knapsack can be interpreted as the total demanded

capacity required to supply all clients in the CFL. The objects in the MKP are possible locations for a facility in the CFL. The capacity of each facility can be interpreted as its value. The opening cost can be interpreted as the weight. Therefore the value and weight of an object are respectively the capacity u and the opening cost o of a facility. In Table 1, the comparison between CFL and MKP is summarized.

Table 1: Comparison between CFL and MKP.

CFL	MKP
Set of facilities $\mathcal{F}$	Set of items/facilities $\mathcal{F}$
Set of clients $\mathcal{D}$	-
Opening cost of a facility, o_i	Cost/weight of an item/facility, o_i
Capacity of a facility, U_i	Capacity of an item/facility, U_i
Assignment cost (facility to client), c_{ij}	-
Demand of a single client, 1	Total demanded capacity, D.

In terms of the capacitated facility location problem, the corresponding minimum knapsack problem is defined as follows. Given a set of possible locations $\mathcal{F}$, where each location $i \in \mathcal{F}$ has a capacity u_i and an opening cost o_i, $i \in \mathcal{F}$. The goal is to choose a subset of locations such that the total capacity of the subset meets a certain demand, D, while the total cost, $C = \sum_{i \in I} o_i y_i$, is as low as possible. Here y_i is the decision to open facility i or not. If facility i is opened, then $y_i = 1$, else $y_i = 0$. Note that the demanded capacity D should be equal to the total demand of all clients. In the CFL investigated in this thesis, each client has demand 1. Therefore, the demand D should be equal to the number of clients $|\mathcal{D}|$: $D = |\mathcal{D}|$. The minimum knapsack problem is shown in Figure 8.

$$\text{minimize} \quad \sum_{i \in I} o_i y_i \tag{16}$$

$$\text{subject to:} \quad \sum_{i \in I} u_i y_i \geq D \tag{17}$$

$$y_i \in \{0, 1\}, \quad \forall i \in \mathcal{F} \tag{18}$$

Figure 8: Minimum Knapsack Problem

An example is shown in terms of the capacitated facility location. A company requires distribution centers to supply all his clients. To meet the demand of all clients, a total capacity of 10 units is required. The company can choose from five locations, each with its own capacity and cost. The goal is to choose a set of locations such that the capacity is greater than or equal to the demanded 10 units, while the cost of the set of chosen locations is minimized. Each location can only be chosen once. Figure 9 shows an example of the situation.

The solution for the specific problem of Figure 9 can easily be found: the two location with cost 4 should be chosen. This yields a total cost of 8 and a capacity of 10, which meets the demand. Due to the small number of locations to choose from, the optimal solution could be found and verified quickly by checking all possible solutions. However, as the number of facilities is increased, the number of possible combinations to check increases exponentially. In each combination, each facility can be either included or excluded. For N facilities, this gives 2^N different possibilities to check. For large N, it is practically impossible to check all combinations. Often it is not necessary to find an optimal solution, however. As long as the found solution

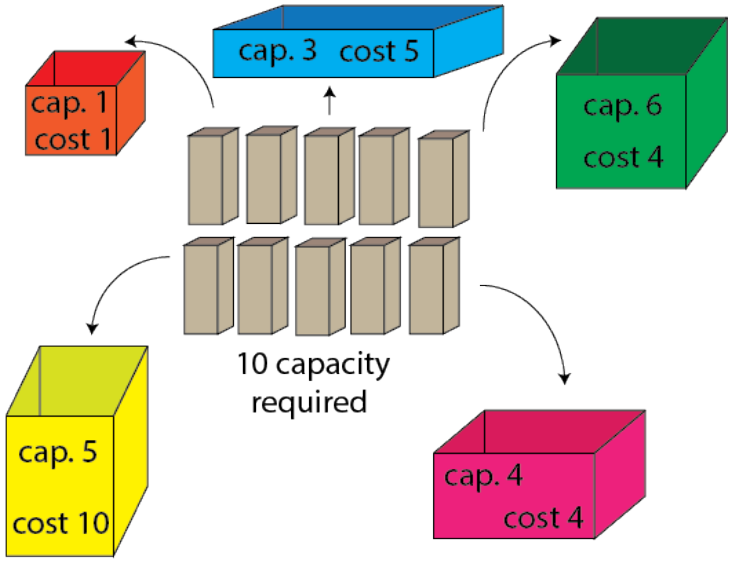


Figure 9: Illustration of the minimum knapsack problem in terms of the CFL. A total capacity of 10 units is required to supply all clients indicated with the rods in the middle. The total demanded capacity can be satisfied by choosing from 5 locations each with corresponding capacity (cap.) and cost. The goal is to minimize the total cost.

approximates the optimal solution within a certain range, the solution will suffice. The goal is to construct an algorithm that finds such a sufficiently good solution in polynomial time, i.e. the time the algorithm requires should scale polynomially or slower with the number of possible locations. Later on, this will be done using a linear relaxation with finite integrality gap. First, a different approach is considered, a greedy algorithm.

3.1.1 Greedy algorithm

A very straightforward method for finding a possible solution is by means of a greedy algorithm. This algorithm finds a solution by first including the location with the highest capacity per unit cost, then the next highest and so on. The time required to order all facilities scales polynomially (at most quadratically) with the number of facilities and the time to check if a subset exceeds the cost scales linearly with the number of facilities. Therefore, the time required for the greedy algorithm scales polynomially with the number of possible facilities. Although this algorithm is very fast, it may also give a very poor result. For example, the demanded capacity $D = 100$, and there are 3 facility locations to choose from: the first location with capacity 99 and cost 1, the second location with capacity 100 and cost 100 and the third location with capacity 1 and cost 2. The locations are summarized in Table 2.

Table 2: Table with object values and costs

Location i	Capacity u_i	Cost o_i	Capacity/Cost, u_i/o_i
1	99	1	99
2	100	100	1
3	1	2	1/2

The greedy algorithm takes the objects with the highest capacity/cost-ratio as long as the total value has not exceeded the demand of 100. Using the greedy algorithm, first location 1

is included, and then location 2. The total capacity of locations 1 and 2 combined is 199, so the demanded value is exceeded and the algorithm stops. This gives a total cost of 101. The optimal solution however is obtained by taking object 1 and 3 for a total cost of 3. The solution found by the greedy algorithm is about 34 times worse than the optimal solution. In general, there is no guarantee for the quality of this greedy algorithm. A good algorithm for the MKP should be bounded for the ratio between the found solution and the optimal solution. The huge gap between the optimal solution and the solution found in the greedy algorithm is because an entire new facility has to be opened when possibly only a small fraction of the capacity of the facilities capacity is required. This gives lead to the approach of interest for this thesis, the linear relaxation of the problem, in which fractional solution are allowed.

3.2 Linear relaxation of the knapsack problem

The difficulty of the knapsack problem lies in the fact that it is an integer problem. A solution can only include ($y_i = 1$) or exclude ($y_i = 0$) a location; for example it is not allowed to include half a location ($y_i = 1/2$). A possibility for approximating the optimal solution is by investigating the linear relaxation of the knapsack problem, which does allow fractions of locations in the solution. The linear relaxation is defined as follows:

$$\text{minimize} \quad \sum_{i \in I} o_i y_i \tag{19}$$

$$\text{subject to:} \sum_{i \in I} u_i y_i \geq D \tag{20}$$

$$y_i \in [0, 1], \quad \forall i \in \mathcal{F} \tag{21}$$

Figure 10: Linear relaxation of the knapsack problem

Again o_i is the opening cost for opening facility i, u_i is the capacity of facility i, D is the total demanded capacity and finally y_i is the decision variable to open facility i. In the linear knapsack problem, there are less restrictions compared to the integer knapsack problem. The linear knapsack problem is a relaxation of the integer minimum knapsack problem. The restriction on $\mathbf{y}$ is relaxed from $y_i \in \{0, 1\}$ in the integer case (Equation (18)), to $y_i \in [0, 1]$ in the linearized case (Equation (21)). Because there are less restrictions for the linear problem, any possible integer solution is also allowed in the linear case, but not the other way around. Therefore the minimum cost of the fractional solution is a lower bound for the minimum cost for the integer solution.

In contrast to the integer MKP, the linearized problem can be solved optimally with the greedy algorithm. The greedy algorithm was not optimal for the integer MKP because the algorithm would include a remaining location with the best capacity/cost-ratio entirely. However, in the linear MKP, a feasible solution can contain fractions of locations. The algorithm includes only the most valuable locations, i.e. the locations with the highest capacity per cost. When the algorithm reaches the required capacity while only including a facility fractionally, the algorithm stops. The solution is feasible, because fractions are allowed in the linear problem. Therefore, the greedy algorithm will not include more locations or part of locations than necessary. Because the most profitable locations are chosen first, the algorithm will always find an optimal solution for the linear problem.

If the same example in Table 2 is reconsidered, then the greedy algorithm would first include location 1 in the solution. Then there is only 1 capacity left to fill up and the algorithm takes

1/100 part of location 2. This yields a total capacity of 100, which equals the demand, for a total cost of $M = 1 + 1/100 \cdot 100 = 2$. Do note that this optimum of the linear problem is lower than the optimum of the integer minimum knapsack problem, which was 3. As explained before, due to the relaxed restrictions, the optimum of the linearized problem is a lower bound for the optimal solution to the integer problem. The integrality gap of the linear relaxation in Figure 10 is not bounded by a constant in general. Later on it will be shown that it is possible to strengthen the linear relaxation, such that the integrality gap is bounded by a constant.

If a linear relaxation has an integrality gap bounded by a constant, a possible way to prove this, is by using an LP-rounding algorithm to convert the fractional optimal solution to a good integral solution. This integral solution is not required to be optimal, so long as its cost approximates the cost of the optimal integral solution. Also it is required to construct such an integral solution in polynomial time, otherwise the construction takes too long for large instances of the problem. If a bound is found for the ratio between the optimal linear solution and the constructed integral solution, this is a bound for the integrality gap as well, as the optimal integral solution has a lower cost than the constructed integral solution. This useful property will be used later on to prove upper bounds for integrality gaps.

First an example is shown for a possible method to construct an integral solution to the minimum knapsack problem from the solution to the linear relaxation. A possible algorithm is to include all fully and partially used locations in the fractional, optimal solution to the knapsack problem. The fractional solution $\mathbf{y}^*$ to the example in Table 2, $\mathbf{y}^* = (1, \frac{1}{100}, 0)$, this would be converted to the integral solution $\overline{\mathbf{y}} = (1, 1, 0)$. Note that the solution to the linear problem had a cost of $C_{lin} = 2$, whereas the integral solution has a cost of $C_{int} = 101$. The gap between the found integral solution and the linear solution is about 50. This algorithm is very simple, but has the disadvantage that it also does not guarantee a good result. In general, the ratio between the optimal linear solution and constructed integral solution is not bounded by a constant. The problem is not due to the algorithm, but due to the chosen relaxation. It will be shown that no algorithm exists which can construct a good integral solution from the fractional solution of the MKP described in Figure 8. This is due to the fact that its integrality gap for the relaxation in Figure 10, is not bounded by a constant, and therefore neither is any constructed integral solution.

The integrality gap for the MKP described in Figure 8 can be as big as the demand D [5]. This limit is reached in the following example. Consider a demand D and two possible facilities, the first has capacity $D - 1$ for zero cost and the second has capacity D for a cost of 1. The fractional solution will first include the facility 1 for free, and fill up the remaining demand, which is 1, with a fraction $1/D$ of the second facility. The linear optimal solution is $y = (1, \frac{1}{D})$, which has a capacity of $D - 1 + \frac{1}{D} \cdot D = D$, so the demand is met. The total cost is $M_{frac} = 1 \cdot 0 + \frac{1}{D} \cdot 1 = \frac{1}{D}$, but the integral solution for this MKP is $(0, 1)$ for a cost of $M_{int} = 1$. Therefore the integrality gap is $IG = \dfrac{C_{int}}{C_{frac}} = \dfrac{1}{1/D} = D$. The integrality gap can be as big as the demand D, and therefore, is not bounded by a constant for a general instance of MKP.

In order to reduce the integrality gap, the linear problem can be strengthened with additional conditions to better approximate an integer scenario. It is possible to add additional constraint that only affect the linearized problem, but are redundant for all integer solutions feasible to the ordinary MKP of Figure 8. In this way, the linear optimal solution will be closer to the optimal integral solution. A possible method for doing so is introduced in the next section.

3.3 Strengthening the Linear Knapsack Problem

Last section showed the original linearized knapsack problem could result in a large integrality gap. By adding more constraints to the linear relaxation of Figure 10, it is possible to reduce the integrality gap significantly. This thesis shows a method introduced by Carr et al. [5], which strengthens the knapsack problem in Figure 10 in general, and in Figure 8 in particular, reducing its integrality gap. An algorithm is used to convert the solution of a strengthened, linear knapsack problem to an integral solution with at most twice the cost of the fractional, optimal solution. This proves that the optimal integral solution is also at most twice as expensive as the linear solution and thus the integrality gap is 2.

In order to improve the integrality gap, first the linear relaxation should be strengthened with additional constraints. These constraints will be redundant for any integral solution, but not for fractional solutions. Therefore, the constraints strengthen the linear relaxation of MKP. Note that the strengthened formulation will still be a relaxation of MKP, because all integer solutions that were initially feasible, will still be feasible when the additional constraints have been added.

The constraints that will be added, force any fractional solution for the full problem to be a solution for any residual set of facilities as well. Consider a subset of facilities $A \subset \mathcal{F}$ such that the total capacity of facilities in A is less than the demanded capacity: $u(A) = \sum_{i \in A} u_i < D$. If all facilities in A are included in a solution, then a residual demand $D(A) = D - u(A)$ still has to be added to meet the demanded capacity. The residual demand must be brought forth from facilities in $\mathcal{F} \setminus A$. Therefore a minimum knapsack problem for the residual subset $\mathcal{F} \setminus A$ with residual demand $D(A)$ has been created. In any knapsack problem, it can be assumed that the capacity of each facility is no higher than the demand: whether a facility has capacity equal to the demand or greater, including the facility will always satisfy the demand. Therefore in the residual problem, all capacities of residual facilities in $i \in \mathcal{F} \setminus A$ are changed such that no facility has capacity greater than the residual demand:

$$u_i^A := \min\{u_i, D(A)\}, \forall i \in \mathcal{F} \setminus A. \tag{22}$$

Note that this specific change strengthens the original problem. Without the adaption of the capacity of each facility, the solution to the original problem would still be allowed. Also note that any integral solution to the ordinary MKP is still feasible when the capacities are adapted: the capacity of the opened residual facilities satisfy demand $D(A)$, because the integral solution was feasible to the ordinary MKP. When the capacity of each facility is adapted to at most $D(A)$, the capacity of each fully opened facility is unchanged (which means the solution is still feasible), or an opened facility now has capacity $D(A)$. Because this facility is integrally opened, the capacity of the residual opened facilities is still $D(A)$ and is therefore satisfies the demand of $D(A)$ and is feasible. Therefore, adding the mentioned constraints does not influence the optimal integral solution.

Table 3: Possible locations and their capacities and costs.

Location i	Capacity u_i	Cost o_i	Capacity/Cost, u_i/o_i
1	99	1	99
2	100	100	1
3	1	2	1/2

For example, reconsider the situation described is Table 2, which is shown again above. The solution to the original linear relaxation was $\mathbf{y}^* = (1, \frac{1}{100}, 0)$. Now an extra condition is added: given that location 1 is part of the solution, the residual part of the solution $\mathbf{y}$ should still meet

the residual demand $D(A) = 100 - 99 = 1$. In other words, $\mathbf{y}' = (0, \frac{1}{100}, 0)$ should still meet the residual demand $D(A) = 1$. If the capacities of residual locations, i.e. location 2 and 3, are not adapted, the residual demand will still be met:

$$\begin{aligned} u(y') &= u^T y' \\ &= 0 \cdot 99 + \tfrac{1}{100} \cdot 100 + 0 \cdot 1 \\ &= 1 \\ &= D(A) \end{aligned}$$

This is also true for a general case, because the capacities in the fractional solution are unchanged and the opening of facilities in A are increased to a full opening of $y_i = 1$, so this will provide enough capacity. However, if the capacity of the residual location are adapted as described by Equation (22), some capacities may be lowered and therefore less solutions are feasible. This becomes very clear when the capacities in this example are adapted. Because the residual demand is only $D(A) = 1$, the capacity of the residual facility 2 has to be changed from 100 to 1 as shown in Equation (22). The situation for the residual problem now changes as shown in Table 4.

Table 4: The situation for the residual subset of location 2 and 3.

Location i	Capacity u_i^A	Cost o_i	Capacity/Cost, u_i^A/o_i
2	1	100	1/100
3	1	2	1/2

With the additional constraint added, adding 1/100 part of location 2 is not sufficient to reach the residual demand of $D(A) = 1$, and the solution to the simple linear relaxation is no longer a feasible solution. The optimal solution for this strengthened relaxation will change. Location 3 will be included entirely in the solution due to its new higher capacity/cost-ratio of 1/2. Therefore, using this single additional constraint, the new total solution becomes $(1, 0, 1)$ for a total cost of 3, instead of $(1, \frac{1}{100}, 0)$ for a cost of 2.

As shown in the example, the additional constraint does strengthen the linear problem. For the final strengthened linear knapsack problem, all subset constraints are added. In other words, any solution for the strengthened linear MKP will have to be feasible for all residual subsets $\mathcal{F} \setminus A$ with corresponding residual demands $D(A)$. The strengthened MKP is shown in Figure 11.

$$\text{minimize} \quad \sum_{i \in I} o_i y_i \tag{23}$$

$$\text{subject to:} \sum_{i \in \mathcal{F} \setminus A} u_i^A y_i \geq D(A), \quad \forall A \subset \mathcal{F} \text{ , with } u(A) < D \tag{24}$$

$$y_i \in \{0, 1\} \qquad \forall i \in \mathcal{F} \tag{25}$$

Figure 11: Strengthened relaxation of MKP

The number of additional constraints added in this way can grow exponentially with the number of facilities $|\mathcal{F}|$, but is finite. The maximum number of possible subsets is $2^{|\mathcal{F}|}$, because each facility can be either included or excluded from each subset. Therefore there are $2^{|\mathcal{F}|}$

possible different combinations. Not all of these are added to the linear MKP though, as the capacity of each subset A should be smaller than the demanded capacity D, $u(A) < D$. Still checking and adding these constraints can take an exponentially long time. This can also be seen when running the *Matlab* script in Appendix A. The first part of this script adds the additional constraints using a for-loop iterating over exponentially many possibilities. However, in practice often only a small number of these additional constraint is sufficient to replace the entire set of constraints to get the same polyhedron. Possibly, the fractional solution can be approximated by use of heuristics. The precise process of determining or approximating the fractional solution in polynomial time is beyond the scope of this thesis however and will not be investigated further. In this thesis and corresponding *Matlab* script, the simplex method was used, which is takes at worst exponentially long, but is in general a fast method. The focus of this thesis lies on the strengthening of the integrality gap between a fractional solution and an integral solution in polynomial time. For the rest of the thesis, the optimal solution to the linearized problem is assumed to be known. In the next section it is shown that the optimal solution to the strengthened linear MKP can be converted to an integral solution which cost no more than twice the cost of the linear solution to the strengthened MKP, $C_{int} \leq 2 \cdot C_{lin,opt}$.

3.4 Integral Minimum Knapsack solution

Given a fractional solution $\mathbf{y}^*$ to the strengthened linear knapsack problem as shown in Figure 11, an algorithm for finding a feasible integral solution is summarized. It is proven that this solution has cost no more than twice the cost of the fractional solution. Because the optimal integral solution to the strengthened MKP has equal to or lower cost than any other integral solution, it follows that the integrality gap of the strengthened MKP, shown in Figure 11, has integrality gap $IG \leq 2$. The algorithm and its proof are summarized as described by Carr, Fleisher, Leung and Phillips [5].

The rounding algorithm starts with an optimal solution $\mathbf{y}^*$ to the strengthened MKP and the following steps are followed to create an integral solution $\overline{\mathbf{y}}$.

1. Fully open all facilities i with original opening half or greater: $y_i^* \geq 1/2 \Rightarrow \overline{y}_i = 1$. Let $I = \{i \in \mathcal{F} : y_i^* \geq 1/2\}$ be the set of facilities with original opening half or greater.

2. Let r be the least common multiple of denominators of all y_i^*, $i \in \mathcal{F} \setminus I$. Create r "buckets", each representing a feasible solution to the residual knapsack problem with possible locations $\mathcal{F} \setminus I$ and demand $D(I)$. Create $a_i := 2r \cdot y_i^*$ copies of each facility $i \in \mathcal{F} \setminus I$.

3. The copies of the facilities in $\mathcal{F} \setminus I$ are now divided over the buckets as follows. Re-index the facilities $i \in \mathcal{F} \setminus I$ in order of decreasing capacity, i.e. the facilities are reordered $y'_1, y'_2, \ldots, y'_n$, such that $u'_1 \geq u'_2 \geq \cdots \geq u'_n$. Put the a'_1 copies of the first reordered facility y'_1 in the first a'_1 buckets, the a'_2 copies of the second facility in the next a'_2 buckets modulo r and so on for all facilities in order of decreasing capacity.

4. All buckets contain feasible solutions to the residual knapsack problem. Evaluate all bucket costs and choose the cheapest bucket. Construct the final solution $\overline{\mathbf{y}}$ by including the facilities in I and the facilities in the cheapest bucket.

The implementation in *Matlab* of this algorithm can be found in Appendix A. The execution of the algorithm takes polynomial time, which is shown by the for-loops used in the script. No for loop in the conversion algorithm scales exponentially with the number of facilities. It will

now be proven that this algorithm yields a feasible integral solution with cost no more than 2 times the cost of the fractional optimal solution. Most importantly, if this algorithm can construct an integral solution at most twice as expensive as the linear optimal solution, then the optimal integral solution is also at most a factor 2 as expensive as the linear optimal solution. Therefore the integrality gap $IG \leq 2$.

Theorem 3.1 (MKP integrality gap)**.** *The strengthened knapsack problem in Figure 11 has integrality gap of at most 2.*

Proof. It will be proven that the algorithm described above can indeed construct an integral solution with cost of at most 2 times the cost of the fractional solution. The optimal integral solution is equally expensive or cheaper than the constructed integral solution. Therefore, the optimal integral solution also has cost of at most $C_{int} \leq 2C_{lin,opt}$. This proof will show the validity of the constructed integral solution and the limit to its cost. In this proof an example will be considered simultaneously for illustration. Consider the optimal solution to the linear relaxation of the strengthened MKP, $\mathbf{y}^* = (y_1, y_2, y_3) = (\frac{2}{3}, \frac{1}{3}, \frac{1}{4})$.

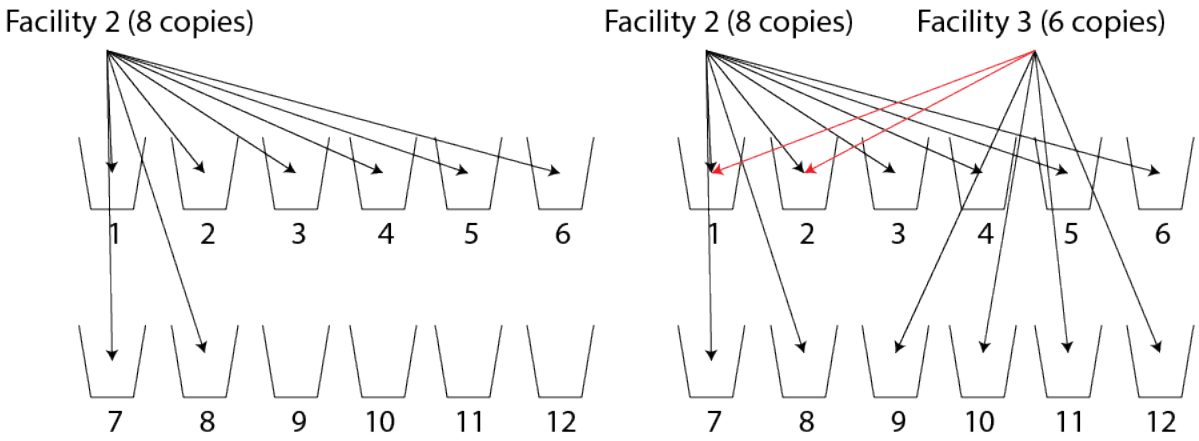
In step 1 of the algorithm, the facilities of half opening or larger are fully opened, $y_i^* \geq 1/2 \Rightarrow \overline{y}_i = 1$. This costs at most twice their contribution to the integral solution. The collection of facilities fully opened this way are defined as $I = \{i \in \mathcal{F} : y_i \geq 1/2\}$. A residual demand $D(I)$ still has to be satisfied with facilities in $\mathcal{F} \setminus I$. In the example, only $y_1^* \geq \frac{1}{2}$, so only facility 1 is opened fully. The intermediate (semi-integral) solution becomes $\hat{\mathbf{y}} = (1, \frac{1}{3}, \frac{1}{4})$.

In step 2, r is the least common multiple of denominators in $\mathcal{F} \setminus I$, so in the example, the denominators of interest are 3 and 4. The least common multiple of 3 and 4 is $r = 12$. Note that all y_i^* have a denominator, because the solution to the linear strengthened MKP always consist of rational numbers. This is because an optimal solution is always on a vertex of the polyhedron created by the constraints. These vertices have fractional coordinates, as all constraints only use rationals. Therefore there is always a rational optimal fractional solution.

Next, r buckets are created and of each facility $i \in \mathcal{F} \setminus I$, $a_i := 2r \cdot y_i^*$ copies are made. In the example, there are $a_2 = 2 \cdot 12 \cdot \frac{1}{3} = 8$ copies of y_2^* and $a_3 = 2 \cdot 12 \cdot \frac{1}{4} = 6$ copies of y_3^*. There are always less copies of a facility than the number of buckets $a_i < r$, because
$y_i^* < \frac{1}{2}, \forall i \in \mathcal{F} \setminus I \Rightarrow a_i := 2r \cdot y_i^* < 2r \cdot \frac{1}{2} = r$.
The facilities in $\mathcal{F} \setminus I$ are re-indexed in order of decreasing capacity u_i. This means that for y_1', y_2', $\ldots$ y_n' are indexed such that $u_1' \geq u_2' \geq \cdots \geq u_n'$. the corresponding number of copies of each re-indexed facility y_i' is a_i'. Then the first a_1' copies of the facility with the highest capacity are distributed over the first a_1' buckets, then the a_2' copies are distributed over the next buckets a_2' buckets modulo r and so on. Because there are less copies of each facility then the number of buckets, no bucket will contain the same facility more than once.

In the example, if facility 2 has higher capacity than facility 3, then facility 2 is re-indexed to $1'$ and facility 3 to $2'$. Then the copies are distributed over the buckets, putting the copies of the facility with highest capacity in the first a_1' buckets and the second highest in the next a_2' buckets modulo r. In the example, the 8 copies of facility 2(= facility $1'$) are put in the first 8 buckets. Then the 6 copies of facility 3(= facility $2'$) are put in the next 6 buckets modulo $r = 12$. So these 6 copies go into bucket 9, 10, 11, 12, 1 and 2. Therefore, the first two buckets contain both facility 2 and 3, bucket 3 up to bucket 8 contain only facility 2 and the last four buckets contain only facility 3. The distribution process for the example is also shown in Figure 12.

The copies were re-indexed from high to low capacity and first the highest ones were distributed over the first buckets, then the second highest ones over the next buckets modulo r and so on. Therefore, the k^{th} copy added to bucket j has greater or equal capacity then the k^{th} copy added to bucket $j + 1$. In Figure 12, this can easily been seen. Consider bucket 8 and 9.



(a) First, the 8 copies of facility 2 are distributed. (b) Second, the 6 copies of facility 3 are distributed.

Figure 12: The distribution of the copies over the buckets as in the example. There are no buckets left for the last two buckets of facility 3, so these are put in in bucket 13 mod 12=1 and bucket 14 mod 12=2. These are marked in red.

The first copy added to bucket 8 was of facility 2, which had greater (or equal) capacity than facility 3, which the first copy added to bucket 9. For all other buckets, the same holds.

Furthermore, the j^{th} bucket contains a greater than or equal to the number of copies than bucket $j+1$. Therefore, bucket j has greater or equal capacity than bucket $j+1$. From induction, it follows that the first bucket contains the highest capacity and the last bucket the lowest. Furthermore, the difference between the highest and the lowest capacity adapted to I, u^I (see Equation 22, A is replaced by I, as now set I is considered). The difference in total adapted capacity u^I between the first and last bucket is at most $D(I)$: in the worst case, the first bucket (highest capacity) has one more facility than the last bucket (lowest capacity). The k^{th} copy in the last bucket was added before the $k+1^{st}$ copy was added to the first bucket and thus has equal or greater capacity. By pairing the k^{th} copy of the last bucket to the $k+1^{st}$ copy in the first bucket, only the first copy in the first bucket is unassigned. Therefore, the difference in adapted capacity u^I is at most the capacity of the first facility added to the first bucket. By definition of the adapted capacity, $u^I \leq D(I)$, thus the difference in capacity between the the bucket with the most capacity and the bucket with the least capacity is at most $D(I)$.

Now it is shown that each of the buckets all contain a feasible solution, i.e. each of the buckets has capacity equal or greater than the required residual demand $D(I)$. If one of the buckets is not feasible, it would have adapted capacity $u^I < D(I)$. If the bucket with the lowest capacity has adapted capacity less than $D(I)$, then all buckets contain less than $D(I)+D(I) = 2D(I)$ capacity, as the difference in adapted capacity u^I between two buckets is at most $D(I)$. Therefore it would follow that the total sum of adapted capacities u^I in the r buckets is less than $r \cdot 2D(I)$. However, the adapted capacity of all $a_i = 2r \cdot y_i$ copies distributed over the buckets was $\sum_{i \in \mathcal{F} \setminus \mathcal{I}} u_i^I a_i = 2r \sum_{i \in \mathcal{F} \setminus \mathcal{I}} u_i^I y_i^*$. The constraints demanded that $\mathbf{y}^*$ would meet the residual demand, $\sum_{i \in \mathcal{F} \setminus \mathcal{I}} u_i^I y_i^* \geq D(I)$. Therefore the total adapted capacity in the buckets is at least $2rD(I)$, but from the assumption it follows that the total capacity is less than $2rD(I)$. Therefore, the assumption is wrong and the bucket with the lowest adapted capacity is at least $D(I)$. Therefore, all buckets cover the residual demand and do not contain any facility twice so all buckets contain a feasible solutions. For the final solution $\overline{\mathbf{y}}$, the cheapest bucket is chosen.

Only the cost of the found integral solution $C_{int} = \mathbf{c}^T \overline{\mathbf{y}^*}$ should be proven to be no more than twice the cost of the initial, fractional solution $M_{frac} = \mathbf{c}^T \mathbf{y}^*$. During the first step in the

algorithm, the cost of the facilities with initial opening half or greater $I = \{i \in \mathcal{F} : y_i^* \geq \frac{1}{2}\}$ is at most doubled. It will be shown that the cost of the cheapest bucket is also no more than twice the cost of the fractional opened facilities $y_i \in \mathcal{F} \setminus I$. In the r buckets, feasible solutions were created and the total cost of all buckets combined is the total cost of all facilities distributed over the buckets. Of each facility $i \in \mathcal{F} \setminus I$, $a_i := 2r \cdot y_i$ copies were created, so the total cost of all buckets is $\sum_{i \in \mathcal{F} \setminus I} o_i a_i = 2r \sum_{i \in \mathcal{F} \setminus I} o_i y_i$. Therefore, the average bucket cost is $2 \sum_{i \in \mathcal{F} \setminus I} o_i y_i$. This means that there exist at least one bucket with cost at most $2 \sum_{i \in \mathcal{F} \setminus I} o_i y_i$, which is twice the cost of the fractional opened facilities $y_i \in \mathcal{F} \setminus I$. Both the opening of facilities in I and opening of facilities in $\mathcal{F} \setminus I$ cost no more than twice the corresponding cost of these facilities in $\mathbf{y}^*$, and therefore the total cost of the constructed integral solution $\overline{y}$ is no more than twice the cost of the fractional optimal solution.

An integral solution with cost twice the optimal fractional solution can always be created given a feasible optimal fractional solution. The optimal integral solution is no more expensive than any other feasible integral solution, therefore the optimal integral solution is also no more than twice as expensive as the linear solution. Thus the integrality gap of the strengthened MKP as described in Figure 11 is at most 2. □

4 Strengthening the CFL relaxation

In this chapter, a strengthening of the relaxation of the CFL is presented. Then the algorithm of An, Singh and Svensson [4] is explained and proven. The fractional, optimal solution of the strengthened relaxation is first converted to a semi-integral solution, in which part of the facilities are fully opened. Second, the semi-integral solution is converted to an integral solution with cost at most 288 times the initial fractional solution. A lot of LP-rounding techniques from the last chapter will be used in the algorithm of An et al. As this thesis is greatly based on the work of An et al., the same notation has been used for consistency and easy comparison.

In order to get a formulation of the CFL with a bounded integrality gap, An et al. have strengthened the linear relaxation of CFL with additional constraints. Starting with the relaxation of the CFL problem with constraints defined in Equations (6)-(7), this formulation was strengthened by means of partial, fractional assignments.

4.1 Partial assignments

A partial fractional assignment is an assignment of clients to facilities such that the capacity of each facility is not exceeded. Clients can be assigned partially, they can also be assigned to multiple facilities. For each facility, the sum of the corresponding assignment variable must be no larger than 1. Formally the conditions for a partial assignment $\mathbf{g} = \{g_{ij} : i \in \mathcal{F}, j \in \mathcal{D}\}$ is shown in Equation (13)

$$\sum_{i \in \mathcal{F}} g_{ij} \leq 1, \quad \forall j \in \mathcal{D} \tag{26}$$

$$\sum_{j \in \mathcal{D}} g_{ij} \leq U_i, \quad \forall i \in \mathcal{F} \tag{27}$$

$$g_{ij} > 0, \quad \forall i \in \mathcal{F}, j \in \mathcal{D} \tag{28}$$

Figure 13: The constraints for a partial assignment $\mathbf{g}$.

Any partial assignment can also be described as a feasible solution to the b-matching polytope of a complete bipartite graph. In this bipartite graph, the set of clients, $\mathcal{D}$, and the set of facilities, $\mathcal{F}$, are the two disjoint sets, and each client can be connected to each facility. Each client can be assigned up to its demand of 1 and each facility can be assigned clients up to its capacity U_i. The bipartite graph is illustrated in Figure 14.

The idea behind the constraints added to the CFL will be that for any feasible partial assignment $\mathbf{g}$, the solution $(\mathbf{x}, \mathbf{y})$ should be able to satisfy the residual demand of each client by means of a flow from each unsaturated client to one or multiple unsaturated facilities. This flow should be be equal to the amount by which client j was not assigned by the partial solution $\mathbf{g}$: $1 - \sum_{i \in \mathcal{F}} g_{ij}$. In other words, no matter how the clients are partially assigned, it should always be possible to assign the remaining demand of the clients using the solution $(\mathbf{x}, \mathbf{y})$ and the partial assignment $\mathbf{g}$ such that each client is fully supplied. Consider the following example with 2 clients and 2 facilities, both facilities have capacity $U_i = 1$. In Figure 15, this example is illustrated.

A possible assignment may be given by connecting $F1 - D1$ and $F2 - D2$. This assignment should also be feasible for any partial assignment. Given a partial assignment allowing $D1$ to get its full demand from $F2$, $F2$ has no capacity left, and $D1$ has no residual demand, but $D2$ still has a demand of 1, which should be routed to an unsaturated facility. In this case $F1$ is

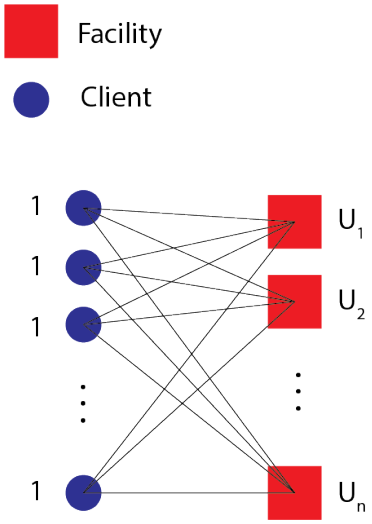


Figure 14: Illustration of the bipartite graph consisting of the disjoint sets $\mathcal{D}$ and $\mathcal{F}$.

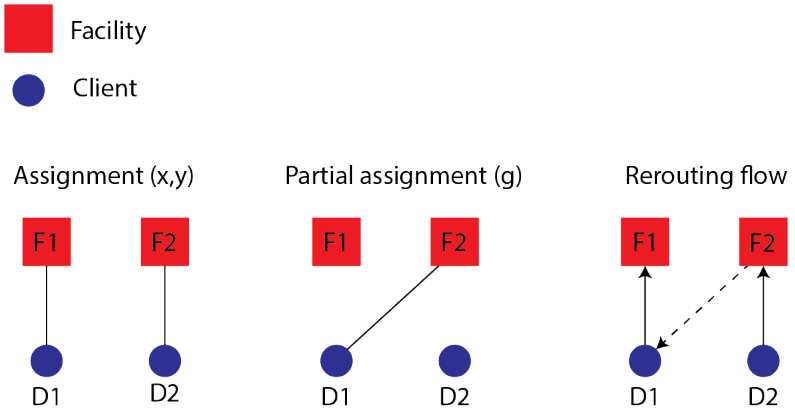


Figure 15: Example for satisfying the residual demand. The assignment $(\mathbf{x}, \mathbf{y})$ should be able to route the residual demand of all clients to unsaturated facilities, such that no facility supplies more clients than its capacity. Flow can be sent over from clients to facilities over arcs x_{ij} and flow can be sent from facilities to clients over arcs g_{ij}. This last flow can be interpreted as reducing the flow initially assigned by the partial solution $\mathbf{g}$.

the only option. The demand of $D2$ can be sent to $F2$ by an integral assignment. $F2$ has no capacity left, but it is possible to reduce the flow $D1 - F2$ by one, thus sending a unit flow from $F2$ to $D1$. Finally the unit flow can be sent from $D1$ to $F1$. The demand of $D2$ is now sent to $F1$ via the route $D2 - F2 - D1 - F1$. As the demand of $D2$ can be satisfied with this integral assignment and partial assignment, the integral assignment may be feasible. To be feasible, such flow paths must exist for all partial assignments.

This technique using partial assignments and residual demands is very similar to the approach used by Carr et al. for the minimum knapsack problem. The MKP was strengthened by demanding that any feasible solution should also be feasible when any subset of facilities is initially opened. A feasible solution to the MKP should then still satisfy the residual demand. For both the CFL and the MKP, it is demanded that given any partial solution, a feasible, total solution must still be able to satisfy the residual demand.

Satisfying the residual demand in the CFL problem has more constraints however than in the MKP. Given a partial assignment $\mathbf{g}$, not only the residual demand of each client must be satisfied by the unsaturated facilities, the facilities also have a limited residual capacity. This must also be taken into account.

$$U_i^g = y_i \left(U_i - \sum_{j \in \mathcal{D}} g_{ij} \right) \tag{29}$$

In Equation (29), the residual capacity U_i^g is the original capacity minus the amount by which the partial assignment connected clients to it. The factor y_i again represents the opening of the facility, which can also be fractional in the linear relaxation.

Furthermore, in Chapter 2 it was also suggested that the constraint $x_{ij} \leq y_i$ could be added to strengthen the linear relaxation. This constraint will also be applied, only slightly adjusted. Instead it will be demanded that a client i can send no more than a fraction y_i of its residual demand to a facility i, i.e. $x_{ij} \leq y_i d_j$.

Summarized, given a partial assignment $\mathbf{g}$ and a feasible solution $(\mathbf{x}, \mathbf{y})$, there are three types of additional constraints: the residual demand of each facility can be satisfied as in Figure 15, each facility does not supply more clients than its residual capacity, and finally, each facility can supply no higher fraction of a client's residual demand than its opening variable y_i. All of these constraints are added together in a multi-commodity flow network.

4.2 Multi-commodity flow network

In the multi-commodity flow network corresponding to the constraint, each client j becomes the source of its own commodity [4]. That is, each client can be interpreted as a commodity, and for each client there exist a source j^s and a sink j^t. It is required that a flow is possible from each client source j^s to its own sink j^t equal to the residual demand of the corresponding client.

Definition 1 (Multi-commodity flow network)**.** *Let $\mathbf{g} = \{g_{ij} : i \in \mathcal{F}, j \in \mathcal{D}\}$ be a feasible partial assignment, and $\mathbf{x} = \{x_{ij} : i \in \mathcal{F}, j \in \mathcal{D}\}$ assignment variables between clients and facilities and $\mathbf{y} = \{y_i : i \in \mathcal{F}\}$ opening variables for all facilities. Let $\mathbf{MFN}(\mathbf{g}, \mathbf{x}, \mathbf{y})$ be a multi-commodity flow network with $|\mathcal{D}|$ commodities, where each client is represented by a source and a sink node, respectively j^s and j^t. Each client source-sink pair is associated with the residual demand of d_j. Each facility is represented by a node pair i and i'. According to the input $(\mathbf{g}, \mathbf{x}, \mathbf{y})$, directed arcs are added to complete the network. Some arcs may have zero capacity.*

1. *For each client source j^s and each facility i, an arc (j^s, i) is added with capacity x_{ij}.*
2. *For each client source j^s and each facility i, an arc (i, j^s) is added with capacity g_{ij}*
3. *For each facility, there is an arc (i, i') with capacity $y_i \cdot (U_i - \sum_{j \in \mathcal{D}} g_{ij})$.*
4. *For each facility and each client sink node, there is an arc (i', j^t) with capacity $y_i d_j$.*

The multi-commodity flow network (MFN) described in the definition is also illustrated in Figure 16. The shaded area contains the arcs (j^s, i) and (i, j^s), which allow clients and facilities to be connected for a feasible solution to the residual problem. This area is the most interesting part of the flow network, as it represents the assignment. The arcs (i, i') and (i, j^t) only exist to represent the residual capacity $y_i \cdot (U_i - \sum_{j \in \mathcal{D}} g_{ij})$ of each facility and the constraint that a facility cannot supply a greater fraction of a clients demand than its opening, $y_i d_j$.

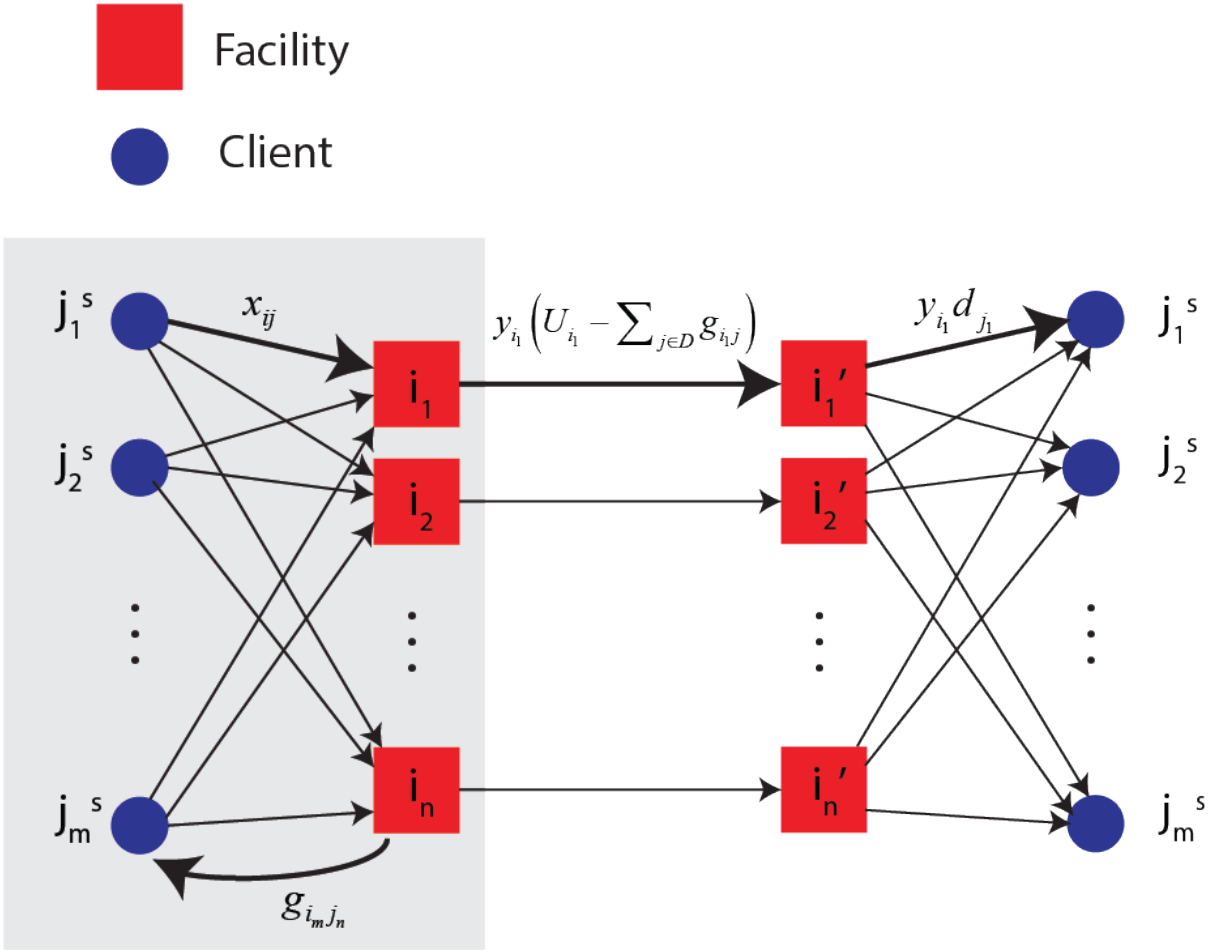


Figure 16: A multi-commodity flow network. The shaded area represents the most essential part of the flow network, the assignment variables. The arcs in the center describe the residual capacity of each facility and the arcs on the right make sure that no client gets a greater fraction of its residual demand from a facility than the opening variable of the facility.

Having defined a partial assignments and the corresponding multi-commodity flow network (MFN), constraints are added to strengthen the linear relaxation of the capacitated facility location problem. The strengthened relaxation will be referred to as MFN-LP:

$$\text{minimize} \quad \sum_{i\in\mathcal{F},j\in\mathcal{D}} c_{ij}x_{ij} + \sum_{i\in I} o_i y_i \tag{30}$$

$$\text{subject to:} \quad \mathbf{MFN}(\mathbf{g},\mathbf{x},\mathbf{y}) \text{ is feasible,} \qquad \forall \mathbf{g} \text{ valid} \tag{31}$$

$$y_i, x_{ij} \in [0,1], \qquad \forall i \in \mathcal{F}, \forall j \in \mathcal{D} \tag{32}$$

Figure 17: MFN-LP, the strengthened relaxation of CFL.

It is not trivial that MFN-LP is indeed a linear program. The constraints added demand that any solution $(\mathbf{x},\mathbf{y})$ should have a feasible $\mathbf{MFN}(\mathbf{g},\mathbf{x},\mathbf{y})$ for every possible partial assignment $\mathbf{g}$. Because the partial assignment is allowed to be fractional, in general there are infinitely many valid partial assignments. However, in Theorem 4.1, it will be shown that it is sufficient to only include partial assignments that are integral for the constraints in Equation (31).

Theorem 4.1 (Integral partial assignments)**.** $\mathbf{MFN}(\mathbf{g},\mathbf{x},\mathbf{y})$ *is feasible for all valid partial assignments* $\mathbf{g}$ *if and only if* $\mathbf{MFN}(\mathbf{g},\mathbf{x},\mathbf{y})$ *is feasible for all valid partial assignments that are integral,* $\hat{\mathbf{g}}$*.*

Proof. Because the set of valid integral partial assignments, $\hat{\mathbf{g}}$, is a subset of the set of all valid partial assignments $\mathbf{g}$, it is trivial that when all $\mathbf{g}$ are valid, then also all integral $\hat{\mathbf{g}}$ are valid. The other way around is harder to prove. First assume that $\mathbf{MFN}(\hat{\mathbf{g}},\mathbf{x},\mathbf{y})$ is feasible for all valid integral partial assignmetns $\hat{\mathbf{g}}$. It will be shown that $\mathbf{MFN}(\mathbf{g},\mathbf{x},\mathbf{y})$ is also valid for all valid fractional partial assignments $\mathbf{g}$.

Any partial assignment $\mathbf{g}$ can be interpreted as weights of the edges in the complete bipartite graph of $\mathcal{D}$ and $\mathcal{F}$, as shown in Figure 14. As $\mathbf{g}$ is valid, no facility is assigned more clients than its capacity, $\sum_{j\in\mathcal{D}} g_{ij} \leq U_i$, and no client is assigned more than its unit demand, $\sum_{i\in\mathcal{F}} g_{ij} \leq 1$. The polytope of feasible solutions corresponding to these constraints has integer valued vertices due to the integralness of U_i and the unit demand of the clients [4]. Therefore it is possible to write $\mathbf{g}$ as a convex combination of valid integral assignments $\hat{\mathbf{g}}^1, \hat{\mathbf{g}}^2, \ldots, \hat{\mathbf{g}}^r$. There exist $\lambda_1, \lambda_2, \ldots, \lambda_r$ such that $\sum_{k=1}^r \lambda_r = 1$ and $\sum_{k=1}^r \lambda_r \hat{\mathbf{g}}^k = \mathbf{g}$. Consider the example in Figure 18. Given 2 clients and 1 facility with capacity 2, a partial assignment may fractionally connect client 1 to facility 1 ($0 \leq x_{11} \leq 1$) and client 2 to facility 1 ($0 \leq x_{12} \leq 1$). A possible partial assignment may be $(x_{11}, x_{12}) = (0.5, 0.75)$. Feasible integral assignments are $(0,0), (1,0), (0,1)$ and $(1,1)$. The fractional partial assignment ($0 \leq x_{12} \leq 1$) can be constructed by taking $(0.5, 0.75) = 0.5 \cdot (1,1) + 0,25 \cdot (1,0) + 0.25 \cdot (0,0)$. This is possible for any feasible partial assignments in any polygon created by the corresponding constraints.

After the partial assignment has been decomposed in terms of integral assignments, consider the feasible flow $\mathbf{f}^k$ of each integral assignment. A flow $\mathbf{f}$ for the partial assignment can be constructed as $\mathbf{f} = \sum_k \lambda_k \mathbf{f}^k$. The flow is feasible due to the linearity of the demands and capacities of the multi-commodity flow network. Also, due to the linearity, the flow satisfies the residual demand of each facility: $d_j(\mathbf{g}) = d_j(\sum_k \lambda_k \hat{\mathbf{g}}^k) = \sum_k \lambda_k d_j(\mathbf{g}^k)$. Each demand $d_j(\mathbf{g}^k)$ was satisfied by the corresponding integral assignment, and thus $d_j(\mathbf{g})$ is also saturated. Therefore, $\mathbf{MFN}(\mathbf{g},\mathbf{x},\mathbf{y})$ is valid for all valid fractional partial assignments $\mathbf{g}$ if $\mathbf{MFN}(\hat{\mathbf{g}},\mathbf{x},\mathbf{y})$ is feasible for all valid integral partial assignments $\hat{\mathbf{g}}$. □

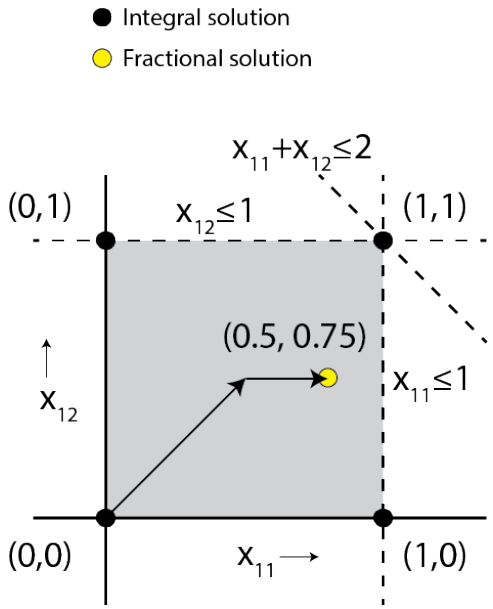


Figure 18: The lines indicate the constraints and the shaded area indicates the feasible region created by the constraints. The fractional solution can be constructed as a linear combination of the integral solutions.

4.3 Validity of the strengthened relaxation

Having defined MFN-LP as the relaxation of the CFL in Equations (30)-(32), it will first be shown that this actually is a relaxation of CFL. To be a relaxation of the CFL problem, any feasible, integral solution to the original CFL problem should also be feasible for the relaxation. This will first be formulated in a theorem, which will be then proven.

Theorem 4.2 (MFN relaxation)**.** *The MFN-LP is a relaxation of the capacitated facility location problem.*

Proof. Given an arbitrary feasible integral solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$, it should hold that for any partial assignment $\mathbf{g}$, $\mathbf{MFN}(\mathbf{g}, \overline{\mathbf{x}}, \overline{\mathbf{y}})$ should be feasible. To prove this, only the integral partial assignments $\hat{\mathbf{g}}$ need to be considered due to Theorem 4.1. Let $\hat{\mathbf{g}}$ be an arbitrary integral partial assignment. The idea behind the proof is to construct flow paths using $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ and $\hat{\mathbf{g}}$, such that a unit flow can be sent from each client with demand not satisfied by the partial solution.

To do so, a directed bipartite graph $G = (V, A)$ is constructed, with all clients $j \in \mathcal{D}$ on one side, and $\overline{y}_i \cdot U_i$ copies of each facility $i \in \mathcal{F}$ on the other side. In other words, if a facility is opened by the solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$, there are U_i copies of the facilities and otherwise none. Figure 19 shows an example of the construction of G and illustrates this proof. Note that for each facility, its capacity is given and whether it is opened or not in $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ in the example. The capacity and the opening determine the number of copies in the matchings below. To construct G, consider the following two matchings, M_1 and M_2.

Matching M_1 has an edge between client j and a copy of facility i if $\overline{x}_{ij} = 1$. This is possible, because $x_{ij} \leq y_i$, so there are copies of facility i. A single copy of a facility will not be allowed to have more than one incident edge, so each client connected with facility i in $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ is sent to a different copy. This is possible since $\sum_{j \in \mathcal{D}} x_{ij} \leq y_i U_i$.

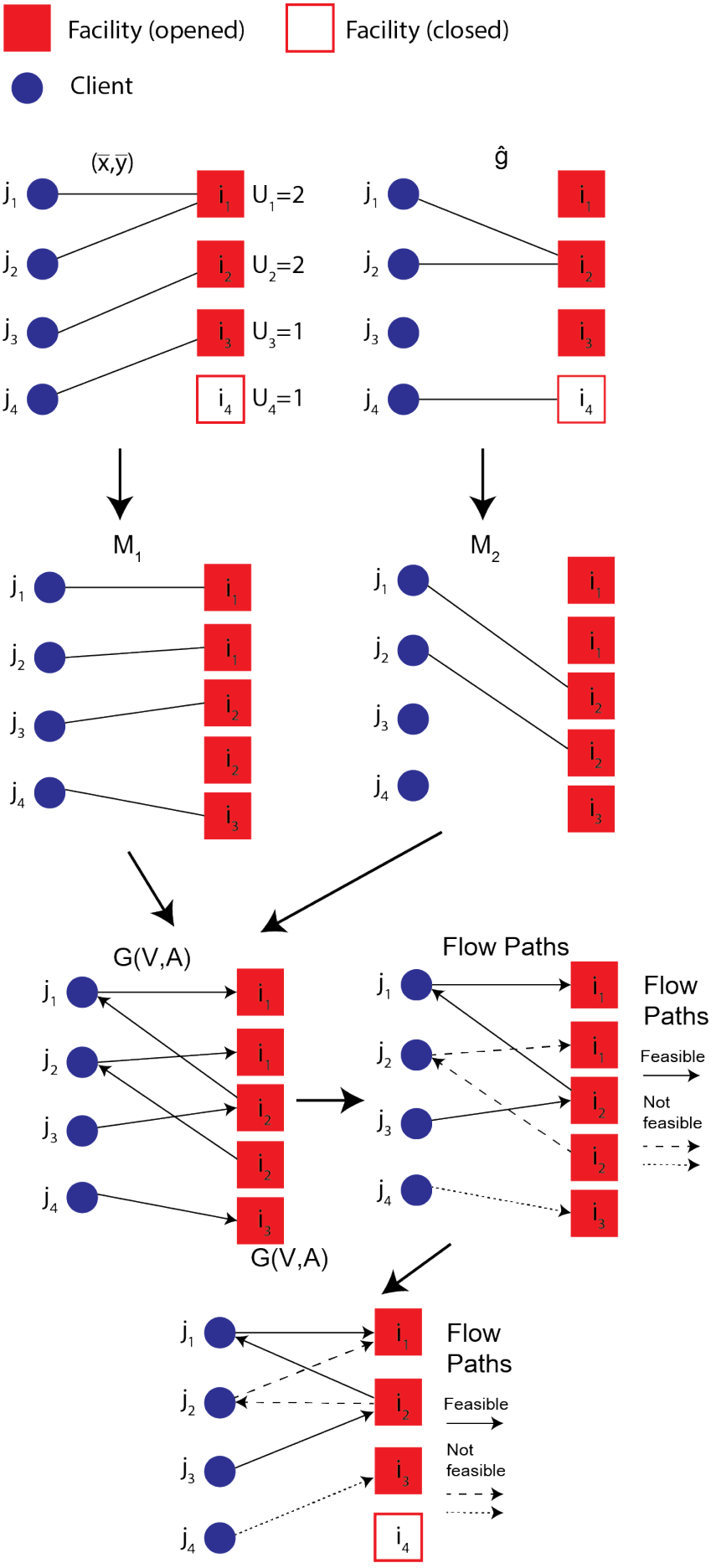


Figure 19: An illustration of the proof for Theorem 4.2. The final flow graph corresponds with the shaded area in $\mathbf{MFN}(\hat{\mathbf{g}}, \overline{\mathbf{x}}, \overline{\mathbf{y}})$.

For matching M_2, there is an edge between a client j and a copy of facility j if $\hat{g}_{ij} = 1$. Again, each client is connected to a different copy, which is possible because $\sum_{j \in \mathcal{D}} g_{ij} \leq U_i$.

Using M_1 and M_2, $G(V, A)$ is constructed, where V stands for the nodes and A for the directed arcs. V is the same set of nodes used in M_1 and M_2, so the set of clients $\mathcal{D}$ plus U_i copies of facility i opened by $\overline{\mathbf{y}}$. Then the set of directed arcs A are added as follows. The arcs (i, j) in M_1 are oriented from client j to facility i and the arcs in M_2 are oriented in the opposite direction. These directed arcs together form A. An example in shown in Figure 19. Because M_1 and M_2 are both matchings, every node in $G(V, A)$ has at most one incoming and one outgoing edge. Therefore, the set of directed arcs A can be decomposed in a set of maximal paths and cycles. The existence of these paths can easily be seen in the figure. A cycle may, for example, occur when both $\hat{\mathbf{g}}$ and $\overline{\mathbf{x}}$ assign a client j to the same facility i. Then there is a directed arc back and forth between i and j, and thus a cycle.

Now ignore the cycles and consider the set $\mathcal{P}$ of paths with nonzero capacity. If a certain path $P \in \mathcal{P}$ starts from a facility or from a saturated client, ignore the path. In the example in Figure 19, the dotted paths are to be ignored. The path starting at j_4 is ignored because j_4 is already saturated by $\hat{\mathbf{g}}$. The path starting from i_2 is ignored because it does not start at an unsaturated client, but at a facility. Only the paths starting from a non-saturated client are of interest, because only these still need to be assigned to a facility. Note that if a client j is not saturated, then $\hat{\mathbf{g}}$ does not assign j to any facility, which means that there is no incoming arc to clients j. For each client, there is exactly one outgoing arc, because $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ assigns each client to a facility. Therefore, each unsaturated client is a starting point of a path $P \in \mathcal{P}$. Because all clients have an outgoing arc, no path can end at a client, so all paths end at a facility i. Now associate each path starting from an unsaturated client j with a flow in the multi-commodity flow network as shown in Figure 16 as follows. First, merge all copies of the same facility and let each directed arc in $G(V, A)$ ending at a copy of a facility i end at the merged corresponding facility i. This is shown in the last step in Figure 19. The path P starting at j and ending at facility i can be denoted as $(j^s, i_1, j_2^s, i_2, \ldots, j_k^s, i)$. Push a unit flow through this path in the $\mathbf{MFN}(\hat{\mathbf{g}}, \overline{\mathbf{x}}, \overline{\mathbf{y}})$. Path P is all part of the shaded area in Figure 16. Then extend this path by adding (i, i', j^t), so the flow is pushed through the facility to the sink node of j. Note that facility i is opened due to the choice of M_1 and M_2. Now facility j is saturated by this flow. If this is done for all non-saturated facilities, a feasible MFN-LP is constructed and the proof is complete. All that is left is to prove that no arc in $\mathbf{MFN}(\hat{\mathbf{g}}, \overline{\mathbf{x}}, \overline{\mathbf{y}})$ is carrying more flow than allowed.

It is easy to prove that no arc in the shaded area in $\mathbf{MFN}(\hat{\mathbf{g}}, \overline{\mathbf{x}}, \overline{\mathbf{y}})$ is carrying more flow than its capacity. All flow paths $P \in \mathcal{P}$ use different arcs, and all arcs in $G(V, A)$ are also present in $\mathbf{MFN}(\hat{\mathbf{g}}, \overline{\mathbf{x}}, \overline{\mathbf{y}})$. Therefore, none of these arcs carry more flow than their capacity. Furthermore, the arcs i', j^t carry a unit flow only if an client with residual demand $d_j = 1$, is connected to an opened facility i by path P. Therefore, the arc i', j^t has capacity $y_i d_j = 1$ and also never carries more flow than its capacity.

Finally only the flows (i, i') still need to be considered. If a facility i was not opened by $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$, the arc (i, i') has zero capacity in $\mathbf{MFN}(\hat{\mathbf{g}}, \overline{\mathbf{x}}, \overline{\mathbf{y}})$. However, because it is closed, this facility was not used in $G(V, A)$. If a facility i was opened by $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$, then it is matched at most $U_i - \sum_{j \in \mathcal{D}} g_{ij}$ in $G(V, A)$. This is because initially there were U_i copies of facility i in M_1, so at most U_i incoming arcs, but there were also $\sum_{j \in \mathcal{D}} g_{ij}$ outgoing arcs from copies of facility i in M_2. Therefore, the highest number of paths P that can possibly end at facility i in $G(V, A)$ is $U_i - \sum_{j \in \mathcal{D}} g_{ij}$. This matches the exact capacity of the arc (i, i') in $\mathbf{MFN}(\hat{\mathbf{g}}, \overline{\mathbf{x}}, \overline{\mathbf{y}})$, and therefore, each of these arcs carries no more flow than its capacity.

Therefore, with the set of extended paths $\mathcal{P}$ all residual demands in $\mathbf{MFN}(\hat{\mathbf{g}}, \overline{\mathbf{x}}, \overline{\mathbf{y}})$ are satisfied and no arc carries more flow than its capacity. Because the integral partial assignment

$\hat{\mathbf{g}}$ was arbitrary, this holds for all valid integral partial assignments. From Theorem 4.1 it follows that $\mathbf{MFN}(\hat{\mathbf{g}}, \overline{\mathbf{x}}, \overline{\mathbf{y}})$ is also feasible for any valid partial assignment. Therefore, any integral solution to the original CFL is feasible to the CFL extended with the MFN constraints. Thus, MFN-LP is a relaxation of CFL. □

4.4 Linear constraints

It has been shown that MFN-LP is indeed a relaxation of the CFL problem. However, it may not be immediately clear why MFN-LP is indeed a linear program. It is far from trivial that the added constraints can be expressed as a linear constraint. In other words, the problem is to write Equations (30)-(32) in a form such as in Figure 5, which is repeated in Figure 20:

$$\begin{array}{ll} \underset{[\mathbf{yx}]}{\text{Minimize:}} & \mathbf{c}^T \cdot [\mathbf{yx}] \\ \text{Subject to:} & \mathbf{A} \cdot [\mathbf{yx}] \leq \mathbf{b} \\ & \mathbf{A}_{eq} \cdot [\mathbf{yx}] = \mathbf{b}_{eq} \\ & \mathbf{0} \leq [\mathbf{yx}] \leq \mathbf{1} \end{array}$$

Figure 20: LP formulation for computationally solving the linear relaxation of CFL.

As was shown in Chapter 2, it is possible to write any solution $(\mathbf{x}, \mathbf{y})$ to the ordinary constraints of CFL in a form $[\mathbf{yx}] = \begin{bmatrix} y_1 & \cdots & y_n & x_{11} & \cdots & x_{n1} & x_{12} & \cdots & x_{nm} \end{bmatrix}^T$ (see Equation 9) with a cost vector $\mathbf{c}$, and the constraints in term of matrices $\mathbf{A}$, $\mathbf{A}_{eq}$ and vectors $\mathbf{b}$, $\mathbf{b}_{eq}$. However, in the constraints in MFN-LP, it is a lot harder to show this can be written in the desired form. The constraints added are that, given a flow network, the capacities of the network, which are dependent on $(\mathbf{x}, \mathbf{y})$, need to be increased such that a minimal multi-commodity flow is possible.

4.4.1 Cutset constraint for single commodity flow

To rewrite these constraints in a desired form for a linear program, the cutsets of the MFN can be considered. The multi-commodity flow network can be converted into a single commodity flow network by adding a general start and sink node. The general source and sink node are respectively connected to each client source and each client sink by a directed arc with capacity d_j, the residual demand of the corresponding client. The arcs are directed from the general source node towards the client sources, and the from the client sinks towards the general sink, as displayed in Figure 21. All the nodes in MFN can then be divided into two disjoint sets, which together contain all the nodes. In the start set, at least the general source node is included, and the sink set includes at least the general sink node. Furthermore, all other nodes are distributed over the start and sink group. Now the cutset of these two sets is the set of arcs that go from the start to the sink group. An example of a possible cutset is shown in Figure 21.

The single-commodity network is feasible if all possible cutsets have at least capacity of the total residual demand [6]. The capacity of each cutset can be expressed in terms of $(\mathbf{x}, \mathbf{y})$ and $\mathbf{g}$. For example, in Figure 21, consider the arcs going out of the start set. These are the arcs (i_1, i_1') with capacity $y_1(U_1 - \sum_{j \in \mathcal{D}} g_{ij})$, the arc (j_2^s, i_2) with capacity x_{22} and arc (j_3^s, i_2) with capacity x_{23}. These arcs must have a combined capacity greater than or equal to the total residual demand: $y_1(U_1 - \sum_{j \in \mathcal{D}} g_{ij}) + x_{22} + x_{23} \geq d_1 + d_2 + d_3$. Note that $\mathbf{g}$ and d_j are all known in the network, so this is indeed a linear constraint dependent on $(\mathbf{x}, \mathbf{y})$. Furthermore, there is only a finite number of possible combinations to include in the start set, and therefore,

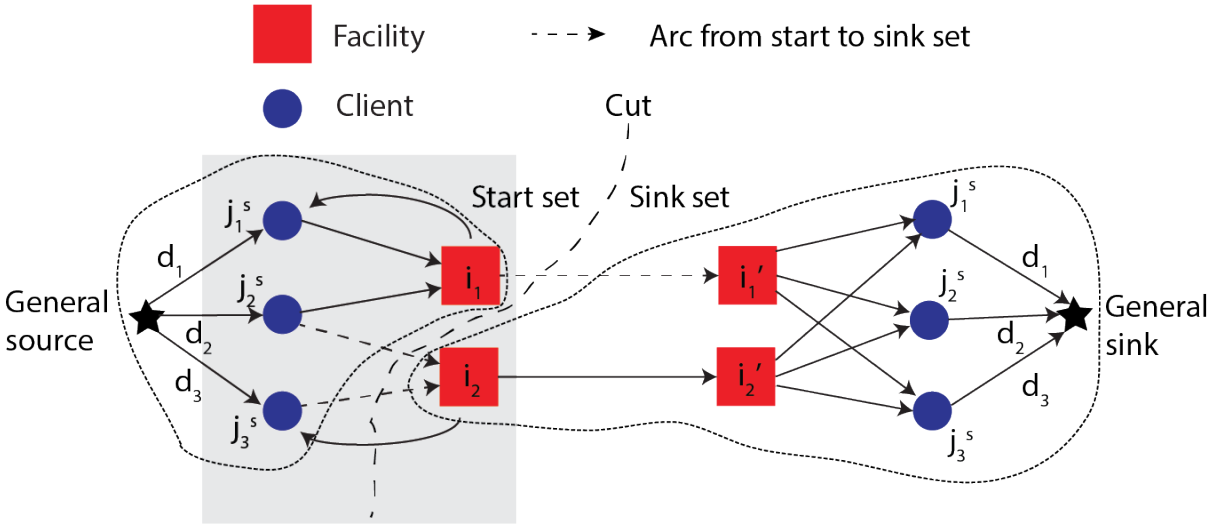


Figure 21: An example of a feasible cutset. The dotted arc are the ones that go from the start set to the sink set.

only a finite number of constraints are added this way.

This approach for adding the MFN-LP constraints is however not sufficient. If a multi-commodity flow network $\mathbf{MFN}(\mathbf{g}, \mathbf{x}, \mathbf{y})$ is possible, that satisfies all residual demands, then there must also be a single-commodity flow network possible that satisfies the total residual demand of all clients. Unfortunately, this is not necessarily true the other way around. In the single-commodity flow network, for example client source 1 can send its residual demand to client sink 2 and the other way around. This is however not allowed in the multi-commodity flow network. In other words, the class of cutset constraints for the single commodity version of each MFN is not sufficient. It is sufficient, however, if it can be proven that for each feasible single commodity version a feasible MFN is possible. It is recommended to try to prove or disprove this in future work. Due to a lack of time, this was not achieved in this thesis. Note that these constraint were added in the *Matlab* script for MFN-LP. If the hypothesis is incorrect, this will have to be changed in the script as well for future use.

4.4.2 Length function constraints

The desired constraints in MFN-LP are that a satisfying multi-commodity flow is possible, where the capacity of the arcs are linearly dependent on the solution $(\mathbf{x}, \mathbf{y})$. An alternative for the cutset-approach is a based on a modified theorem based on a proven theorem of Schrijver [7]. The modified theorem below is identical to the one of Schrijver, except that notation has been synchronized with this thesis and the theorem is specified for MFN.

Theorem 4.3. *The directed fractional multi flow problem* $\mathbf{MFN}(\mathbf{g}, \mathbf{x}, \mathbf{y})$ *has a solution if and only if*

$$\sum_{j \in \mathcal{D}} d_j \cdot \text{dist}_l(j^s, j^t) \leq \sum_{e \in E} l(e) cap(e)$$

for each length function $l : E \to \mathbb{Z}_+$.

Here e denotes a directed arc in E, the set of directed arcs in $\mathbf{MFN}(\mathbf{g}, \mathbf{x}, \mathbf{y})$. $cap(e)$ denotes the capacity of are e. dist_l denotes the shortest distance between a source-sink pair (j^s, j^t),

given the length function l. The idea behind the theorem is that sum of distance each demand has to travel, there should be enough capacity to transport the residual demand over this length.

Adding such constraints for MFN-LP has not been researched thoroughly in this thesis, only the difficulties of this approach are discussed and suggestions are made.

Given a length function l, it is possible to first determine the minimum distance of each sink-source pair, $\text{dist}_j(j^s, j^t)$, and the residual demand is also known. The left hand side is thus a constant. $cap(e)$ is a linear function of $bf(x, y)$, and therefore, this constraint can easily be expressed is a form $\mathbf{a}[\mathbf{yx}] \leq b$, where $\mathbf{a}$ is a row vector in A, and b is an element of $\mathbf{b}$ in Figure 20. The main problem is that there are infinitely many possible length functions l, and therefore, infinitely many constraints to be added. An unproven hypothesis is that only the length functions $l : E \to \{0, 1\}$need to be considered, of which there is only a finite number of arcs. It is possible to write every length function as a linear combination of this set of length functions. For the hypothesis to be correct, it must hold that if Theorem 4.3 is true for all length function $l : E \to \{0, 1\}$, then it should also be true for all nonnegative, linear combinations of these length functions. This is suggested to be proven or disproved in future research. If the hypothesis is correct, the MFN-LP can be expressed in the mentioned linear constraints.

5 288-approximation algorithm

In this section it will be shown that the MFN-LP formulation of the CFL has an integrality gap of at most 288. This will be proven by constructing an integral solution from the optimal, fractional solution to MFN-LP, with at most 288 times the cost of the fractional optimal solution to MFN-LP. In this LP-rounding algorithm, first the fractional solution $(\mathbf{x}^*, \mathbf{y}^*)$ is converted to a semi-integral solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$. Second, the semi-integral solution is converted to an integral solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$. The main theorem of this thesis is formally stated, which was proven by An et al. [4].

Theorem 5.1. *The integrality gap of the MFN-LP is at most 288.*

Theorem 5.1 is proven in this section. The proof consist of two parts, first, the approximation algorithm rounds the fractional solution to a semi-integral solution for a cost of at most 8 times the cost of the fractional solution. Second, the semi-integral solution is converted to an integral solution with a cost of at most 36 times the cost of the semi-integral solution. Therefore, the full algorithm constructs an integral solution of at most $8 \cdot 36 = 288$ times the cost of the fractional solution to MFN-LP.

5.1 From fractional to a semi-integral solution

In this section the first step of the proof for the 288-integrality gap is shown. In this step, the optimal, fractional solution to MFN-LP is rounded and further converted to a semi-integral solution. First, a semi-integral solution is defined [4].

Definition 5.2. *A solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ is semi-integral if it satisfies all of the following conditions:*

1. *$(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ satisfies the assignment constraints, i.e. all clients are saturated, $\sum_{i \in \mathcal{F}} \hat{x}_{ij} = 1$, $\forall j \in \mathcal{D}$, and no facility supplies more than its capacity, $\sum_{j \in \mathcal{D}} \hat{x}_{ij} \leq \hat{y}_i U_i$, $\forall i \in \mathcal{D}$.*

2. *Each facility $i \in \mathcal{F}$ is either integrally opened $y_i = 1$ or at most half opened $y_i \leq \frac{1}{2}$. Let I be the set of integrally opened facilities, $I = \{i : \hat{y}_i = 1\}$, and S be the set of remaining facilities, $S = \mathcal{F} \setminus I$.*

3. *Let the residual demand of a client be the amount client j still demands after being supplied by the integrally opened facilities I. This is the amount client j receives from facilities in S. The residual demand is given by $\hat{d}_j = 1 - \sum_{i \in I} \hat{x}_{ij} = \sum_{i \in S} \hat{x}_{ij}$. A client can get no greater fraction of its residual demand from a facility than the opening of that facility: $\hat{x}_{ij} \leq \hat{y}_i \hat{d}_j$, $\forall j \in \mathcal{D}$, $i \in \mathcal{S}$.*

Having defined a semi-integral solution, it will be shown that the optimal, fractional solution to MFN-LP can be converted to a semi-integral solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ with cost no more than 8 times the cost of the fractional solution $(\mathbf{x}^*, \mathbf{y}^*)$ and in polynomial time.

Theorem 5.3. *An algorithm exist to convert a fractional solution $(\mathbf{x}^*, \mathbf{y}^*)$, feasible to MFN-LP, to a semi-integral solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$. The cost of the semi-integral solution is at most 8 times the cost of the fractional solution: $c(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \leq 8c(\mathbf{x}^*, \mathbf{y}^*)$. Furthermore, this algorithm can be executed in polynomial time.*

The proof of Theorem 5.3 is given in the rest of this section. The proof consist of many steps, each of which is explained and proven to lead to the desired semi-integral solution. The algorithm is summarized in Figure 18. The algorithm will be explained step by step. Also, the

Input: The optimal, fractional solution $(\mathbf{x}^*, \mathbf{y}^*)$ to MFN-LP.
Output: A semi-integral solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$.

1. Let $y'_i = \begin{cases} 1 & \text{if } y^*_i \geq \frac{1}{4}, \\ y^*_i & \text{otherwise} \end{cases}$, for each facility $i \in \mathcal{F}$. Let $I := \{i \in \mathcal{F} : y'_i = 1\}$ be the set of integrally opened facilities and $S = \mathcal{F} \setminus I$ the residual set of facilities.

2. Construct a complete bipartite graph $G = (\mathcal{D}, I, E)$ from the set of clients $\mathcal{D}$ on one side and the set of opened facilities I on the other. E is the set of possible arcs. Each arc (j, i) has a capacity of $2x_{ij}$ (can also be zero). Each client j has a capacity of 1 and each facility $i \in I$ has a capacity of U_i.

3. Find the maximum capacitated b-matching on the bipartite graph G and let $\mathbf{z}$ denote this matching.

4. Let H denote the directed graph associated to the support of the residual network of G. H includes the unsaturated arcs from $\mathcal{D}$ to I and the arcs from I to $\mathcal{D}$ when an arc is used for a connection greater than zero, as to represent that it is possible to reduce the assignment between j and i. Formally stated, $H = \{(j, i) : z_{ij} < 2x^*_{ij}\} \cup \{(i, j) : z_{ij} > 0\}$.

5. Let a client j be called saturated if it its full demand is met by the matching $\mathbf{z}$, $\sum_{i \in \mathcal{F}} z_{ij} = 1$, and unsaturated otherwise. Let I_H and D_H respectively denote the set of client and the set of facilities that are reachable by an unsaturated client:

$$I_H = \{i \in I : i \text{ is reachable in } H \text{ from any unsaturated client}\};$$
$$\mathcal{D}_H = \{j \in \mathcal{D} : i \text{ is reachable in } H \text{ from any unsaturated client}\}.$$

6. Create an integral partial assignment g^*_{ij} as follows:

$$g^*_{ij} = \begin{cases} z_{ij} & \text{if } i \in I_H \\ z_{ij} & \text{if } i \in I \setminus I_H,\ j \in \mathcal{D} \setminus \mathcal{D}_H \\ 0 & \text{if } i \in I \setminus I_H,\ j \in \mathcal{D}_H \\ 0 & \text{if } i \in S \end{cases}$$

7. Let $\mathbf{f}$ be the flow feasible to $\mathbf{MFN}(\mathbf{g}^*, \mathbf{x}^*, \mathbf{y}^*)$, and $h(i, j)$ and $h(S, j)$ respectively the amount of flow sent from j through arc (i, i') and through arcs $(i, i'), i \in S$. The integral semi-integral solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ becomes:

$$\hat{y}_i = \begin{cases} 1 & \text{if } i \in I \\ 2y^*_i & \text{if } i \in S \end{cases}$$

$$\hat{x}_{ij} = \begin{cases} g^*_{ij} & \text{if } i \in I, j \in \mathcal{D} \\ d_j \dfrac{h(i, j)}{h(S, j)} & \text{if } i \in S, j \in \mathcal{D} \end{cases}$$

Figure 22: The LP-rounding algorithm to convert a fractional solution to a semi-integral solution as proposed by An et al. [4].

algorithm is also implemented in a *Matlab* script in Appendix B.

In the first step of the algorithm, the facilities with large openings are fully opened. This happens in a similar way as in the minimum knapsack problem: given that a facility is opened for a certain fraction or greater in the fractional solution, it will be opened fully in the semi-integral solution. In the rounding algorithm used for MKP, a facility was opened if $y_i^* \geq \frac{1}{2}$. However, in the rounding algorithm for the CFL, a facility is already fully opened when $y_i^* \geq \frac{1}{4}$. An intermediate opening y_i' is defined, which will be used to construct the semi-integral solution later on.

$$y_i' := \begin{cases} 1, & \text{if } y_i^* \geq \frac{1}{4}; \\ y_i^* & \text{otherwise.} \end{cases} \tag{33}$$

Each facility is opened for at most 4 times its original opening, so this step makes the opening cost at most 4 times higher than the initial opening cost $c(\mathbf{y}') \leq 4c(\mathbf{y}^*)$ of the fractional solution $(\mathbf{x}^*, \mathbf{y}^*)$ and can be done in polynomial time. Now let I be the set of integrally opened facilities: $I = \{i \in \mathcal{F} : y_i' = 1\}$, and S the set of facilities with zero or small fractional opening, $S = \mathcal{F} \setminus I$. The algorithm opens all facilities in I, and the idea is to partially assign as many clients as possible to these facilities, under condition that the partial assignment cost will not grow too large.

5.1.1 Assigning clients to opened facilities

A partial assignment $\mathbf{g}^*$ is desired, which assigns as many clients as possible to the set of opened facilities I. To ensure this partial assignment will not be too expensive, each client will be allowed to receive at most twice its current assignment from each facility, $2x_{ij}^*$. This can again be represented by a bipartite graph consisting of all clients $j \in \mathcal{D}$ with demand 1 on one side and the opened facilities $i \in I$ with capacity U_i on the other. Each client is allowed to receive twice its current assignment, so between each client $j \in \mathcal{D}$ and each facility $i \in I$, there is an arc with capacity $2x_{ij}^*$. Finding the maximum fractional b-matching of this bipartite graph corresponds to assigning as many clients as possible to facilities in I under the condition that the corresponding partial assignment $\mathbf{g}^*$ (the maximum b-matching) is not too expensive. This ensures that the assignment cost is no more than 2 times the cost of the fractional assignment cost $c(\mathbf{g}^*) \leq 2c(\mathbf{x}^*)$.

Formally the bipartite graph can be denoted as $G = (\mathcal{D}, I, E)$, where $\mathcal{D}$ consists of all clients, I consist of the integrally opened facilities and E are the arcs with capacity $2\mathbf{x}^*$. In the bipartite graph G, each client $j \in \mathcal{D}$ is associated with a capacity of 1 and each facility has a capacity of U_i. Let $\mathbf{z}$ denote the maximum matching of G, then $\mathbf{z}$ maximizes the assignment of clients to facilities in I, $\sum_{i \in I, j \in \mathcal{D}} z_{ij}$, under the conditions:

$$z_{ij} \leq 2x_{ij}^*, \quad \forall i \in I, j \in \mathcal{D}, \tag{34}$$

$$\sum_{j \in \mathcal{D}} z_{ij} \leq U_i, \quad \forall i \in I, \tag{35}$$

$$\sum_{i \in I} z_{ij} \leq 1, \quad \forall j \in \mathcal{D}. \tag{36}$$

Figure 23: Maximum assignment conditions for $\mathbf{z}$.

Equations (34), (35) and (36) respectively denote the capacity of the arcs, $2x_{ij}^*$, the capacity of each facility, U_i, and the demand of each client, 1. As always, facilities can not supply more

clients than their capacity and each client only needs to be assigned up to its demand of 1. This problem can be interpreted as a single commodity maximum flow problem, by adding a source node s and a sink node t. Then add directed arcs from the source node s to each client $j \in \mathcal{D}$ with capacity 1. Also add arcs from each facility $i \in I$ to the sink node t. Each arc $(j, i) \in E$ becomes a directed arc from j to i. The bipartition $G = (\mathcal{D}, I, E)$ and the corresponding single-commodity flow network are illustrated in Figure 24.

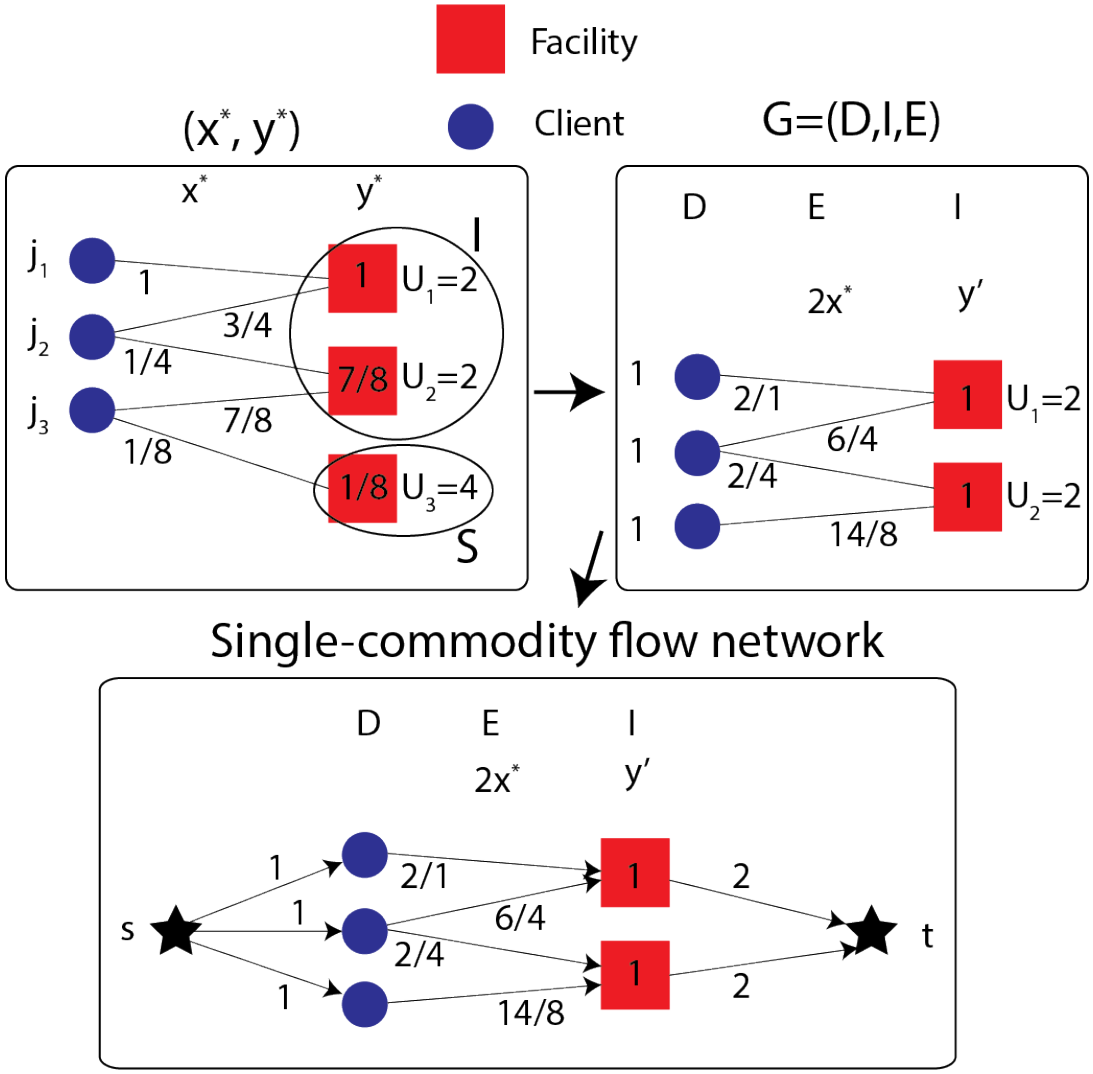


Figure 24: Example of the rounding of the facilities and the conversion to a flow graph. Facility 1 and 2 are integrally opened in $\mathbf{y}'$, as their opening is $y_i \geq 1/4$. Facility 3 is part of $S = \mathcal{F} \setminus I$, because $y_3 < 1/4$. In the upper right corner, $G = (\mathcal{D}, I, E)$ is constructed from $(\mathbf{x}^*, \mathbf{y}^*)$, by opening and including the set I and adding arcs with capacity $2\mathbf{x}^*$. The maximum b-matching of $(\mathcal{D}, I, E)$ can be found by finding the maximum flow in the single-commodity flow network corresponding the bipartite graph $G = (\mathcal{D}, I, E)$. Finding the maximum matching (or maximum flow) corresponds to assigning as many clients as possible to integrally opened facilities.

The max-flow problem can be solved in polynomial time, for example using the Ford-Fulkerson algorithm [6]. Having found the solution to the max flow problem, this corresponds to a maximal matching $\mathbf{z}$ of $G = (\mathcal{D}, I, E)$. In other words, z_{ij} is equal to the amount of flow going from client j to facility i in the corresponding flow network. Using the maximal matching

$\mathbf{z}$, the residual flow network can be constructed. This residual network contains all the directed arcs that are not fully saturated by $\mathbf{z}$. Also the arcs that transport flow are included, but in reverse direction. Sending flow in the reverse direction can be interpreted as a reduction in the flow initially sent in the forward direction. Formally the residual network, H, is defined as follows. H is a directed graph with nodes $\mathcal{D} \cup I$. The set of directed arcs is:

$$\{(j,i) : z_{ij} < 2x^*_{ij}\} \cup \{(i,j) : z_{ij} > 0\} \tag{37}$$

An example of the construction of the residual flow network H is shown in Figure 25. Note that the set of directed arcs is different from the directed arcs in the shaded area of Figure 16. Also $\mathbf{z}$ is not yet the final partial assignment that will be used to continue the conversion algorithm.

To construct the partial assignment, some useful properties of the maximal matching $\mathbf{z}$ will be used. A client $j \in \mathcal{D}$ will be called *saturated* if it is matched precisely once in $\mathbf{z}$: $\sum_{i \in I} z_{ij} = 1$. If a client is not saturated, it is called *unsaturated*. A facility $i \in I$ is called *saturated* if it is fully assigned in the maximal matching $\mathbf{z}$: $\sum_{j \in \mathcal{D}} z_{ij} = U_i$ and *unsaturated* otherwise. Because $\mathbf{z}$ is a maximal matching, there is no path in the residual flow network H from an unsaturated client to an unsaturated facility. Otherwise, an additional flow could be pushed along this path, which would yield a higher total flow, and thus also a greater corresponding matching. This is not possible, since the matching $\mathbf{z}$ is already maximal.

Now consider the set of unsaturated clients and the residual flow network H. The unsaturated clients are of interest, because their demand still has to be satisfied. Using the residual flow network H, it is possible to find out to which clients and integrally opened facilities this residual demand can still be routed. A set of *reachable* clients, $\mathcal{D}_H$, and *reachable* facilities, I_H, are defined as follows [4]:

$$I_H := \{i \in I : \ i \text{ is reachable in } H \text{ from some client } k \text{ that is unsaturated}\} \tag{38}$$

$$\mathcal{D}_H := \{j \in \mathcal{D} : \ j \text{ is reachable in } H \text{ from some client } k \text{ that is unsaturated}\} \tag{39}$$

Note that when a client or facility is called *reachable*, this is a abbreviation of *reachable by some unsaturated client in the residual network H*. Facilities $i \in I \setminus I_H$ and clients $j \in \mathcal{D} \setminus \mathcal{D}_H$ will be referred to as *unreachable*. See Figure 25 for an illustration of reachable and unreachable clients and facilities.

The following lemmas about reachable clients and facilities will prove useful later on

Lemma 5.4. *For the residual network H corresponding to a maximum b-matching $\mathbf{z}$ of $G = (\mathcal{D}, I, E)$, the following must hold:*

(i) Any reachable facility $i \in I_H$ is saturated, i.e., $\sum_{j \in \mathcal{D}} z_{ij} = U_i$.

(ii) If a facility is unreachable, $i \in I \setminus I_H$, but a client is reachable $j \in \mathcal{D}_H$, then the client and facility are at most connected: $\mathbf{z}$: $z_{ij} = x^*_{ij}$.

(iii) If a facility is reachable $i \in I_H$, but a client is not, $j \in \mathcal{D} \setminus \mathcal{D}_H$, then the client and the facility are not connected: $z_{ij} = 0$.

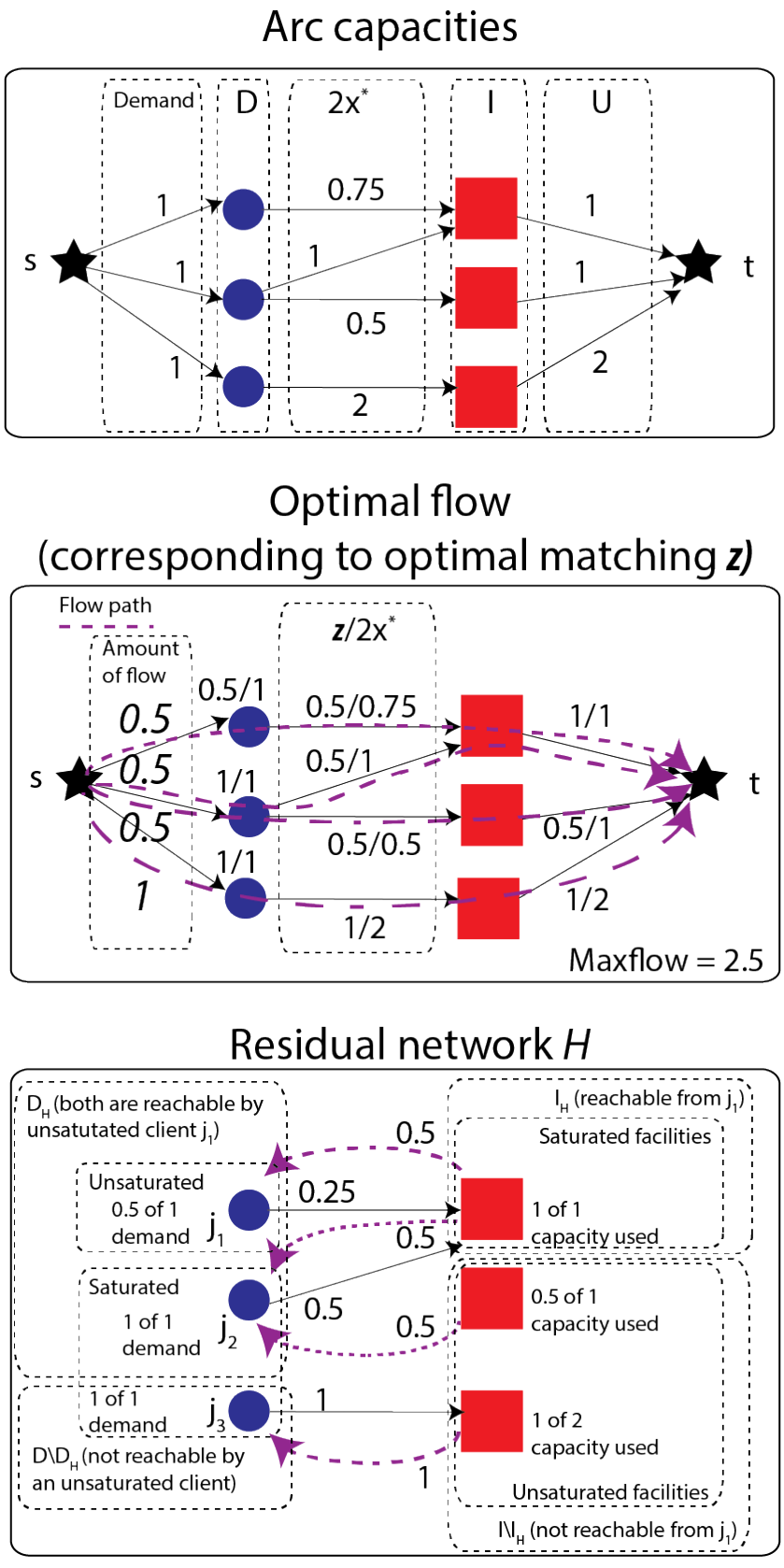


Figure 25: Example of the construction of a residual network. This example is different from the one in Figure 24. In the upper figure, some flow network $(\mathcal{D}, I, E)$ is shown. The middle picture shows the flow paths corresponding to the solution to the max flow problem. The lower picture indicates the residual network H and the set of unsaturated clients (in this case only j_1). Note that an arc is going from j_1 to the upper facility and from there to client j_2. Therefore all of these are reachable by an unsaturated client.

Proof. Note the capacities of the arcs of the residual network H (Equation (37)), as the proof is based on H and the maximal b-matching $\mathbf{z}$.

(i) Any reachable facility $i \in I_H$ must be saturated, otherwise, an additional flow is possible. Facility i is reachable from an unsaturated facility, which can send flow to i. If the facility is not saturated, the flow can be pushed on to the end node, and a greater flow is possible. This contradicts with the fact that $\mathbf{z}$ was a maximal matching.

(ii) Because client $j \in \mathcal{D}_H$ is reachable from an unsaturated client, but facility $i \in I \setminus I_H$ is not, no arc (j, i) can exist in the residual flow network H. Otherwise facility j could be reachable from client i, and thus also from a certain unsaturated client. The arc (j, i) has zero capacity in H only if it is already fully in use by the assignment $\mathbf{z}$: $z_{ij} = 2x^*_{ij}$.

(iii) This proof goes similar to the proof of (ii). Because facility $i \in I \setminus I_H$ is reachable from an unsaturated client, but client $j \in \mathcal{D}_H$ is not, no arc (i, j) can exist. Therefore $z_{ij} = 0$ must hold.

□

Finally the valid partial assignment $\mathbf{g}^*$ is constructed from the maximal b-matching as follows:

$$g^*_{ij} = \begin{cases} z_{ij} & \text{if } i \in I_H \\ z_{ij} & \text{if } i \in I \setminus I_H, j \in \mathcal{D} \setminus \mathcal{D}_H \\ 0 & \text{if } i \in I \setminus I_H, j \in \mathcal{D}_H \\ 0 & \text{if } i \in S \end{cases} \tag{40}$$

The chosen partial assignment $\mathbf{g}^*$ is identical to the maximal b-matching $\mathbf{z}$, except that it leaves out the connections (i, j) between unreachable facilities $i \in I \setminus I_H$ and reachable clients $j \in \mathcal{D}_H$. Note that $\mathbf{g}^* \leq \mathbf{z} \leq 2\mathbf{x}^*$, therefore the cost of $\mathbf{g}^*$ is at most twice the fractional assignment cost: $c(\mathbf{g}^*) \leq c(2\mathbf{x}^*)$. Furthermore, $\mathbf{g}^*$ only assigns clients to integrally opened facilities $i \in I$.

Partial assignment $\mathbf{g}^*$ is a valid partial assignment. This is because $\mathbf{z}$ still satisfies the partial assignment constraints of Equation (13), therefore so does $\mathbf{g}^*$. The multi-commodity flow with the original fractional solution $(\mathbf{x}^*, \mathbf{y}^*)$, $\mathbf{MFN}(\mathbf{g}^*, \mathbf{x}^*, \mathbf{y}^*)$ is therefore feasible. The intermediate opening $\mathbf{y}'$ has greater or equal openings compared to y^*, and therefore, $\mathbf{MFN}(\mathbf{g}^*, \mathbf{x}^*, \mathbf{y}')$ is also feasible as the change from $\mathbf{y}^*$ to $\mathbf{y}'$ only increases some arc capacities.

5.1.2 Assigning clients to fractional opened facilities

Now the following is claimed: there exists a feasible flow to $\mathbf{MFN}(\mathbf{g}^*, \mathbf{x}^*, \mathbf{y}')$ such that at least half of the residual demand of each client is sent to facilities in S. This claim is further described. Let $\mathbf{f}$ be a multi-commodity flow feasible to $\mathbf{MFN}(\mathbf{g}^*, \mathbf{x}^*, \mathbf{y}')$. Let $\mathcal{P}_{ij}$ denote the set of flow paths that carry flow from j_s to j_t using arc (i, i'). These are the flow paths that take flow from client j and sink it at facility i. An example is shown in Figure 26. Let $P \in \mathcal{P}$ be such a flow path, then let $f(P)$ denote the flow carried over this flow path. For each $i \in I, j \in \mathcal{D}$, let $h(i, j)$ denote the total amount of flow carried from client j and sinked at facility i by path $P \in \mathcal{P}_{ij}$, i.e. $h(i, j) = \sum_{P \in \mathcal{P}_{ij}} f(P)$. Furthermore, let $h(X, j) := \sum_{i \in X} h(i, j)$ denote the total amount of flow client j sinks at facilities in X. Formally the claim is stated in Theorem 5.5:

Theorem 5.5. *There exist a feasible flow to* $\mathbf{MFN}(\mathbf{g}^*, \mathbf{x}^*, \mathbf{y}')$ *such that each client* j *sends at least half its residual demand* d_j *to facilities in* S*:* $h(S, j) \geq \frac{1}{2} d_j = \frac{1}{2}(1 - \sum_{i \in \mathcal{F}} g^*_{ij})$.

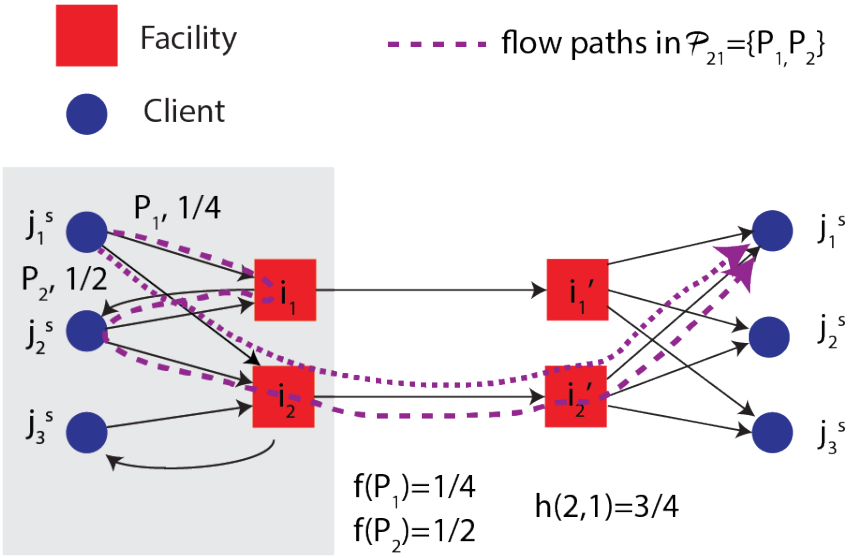


Figure 26: An example of part of a solution to an MFN. The dotted lines indicate all the flow paths (P_1 and P_2) transporting flow from client j_1 to facility i_2. These paths are elements of $\mathcal{P}_{21}$, i.e. the paths transporting flow from j_1 to i_2. The total flow transported $h(2,1) = 3/4$.

The full proof of this theorem can be found in the paper of An, Singh an Svensson. The proof is very extensive and instead this thesis will only focus on the main idea behind the proof. First the construction of the semi-integral solution will be finished, as only one step remains.

For now, it is assumed that Theorem 5.5 is true. Let $\mathbf{f}$ be a flow that sends at least half of the residual demand of each client $j \in \mathcal{D}$ in $\mathbf{MFN}(\mathbf{g}^*, \mathbf{x}^*, \mathbf{y}')$ to facilities in S, note that $\mathbf{f}$ does not necessarily saturate every client, and therefore $\mathbf{f}$ is in general not a valid solution for the multi-commodity flow $\mathbf{MFN}(\mathbf{g}^*, \mathbf{x}^*, \mathbf{y}')$. A feasible flow $\mathbf{f}$ can be found by solving for the maximum flow in the multi-commodity flow network $\mathbf{MFN}(\mathbf{g}^*, \mathbf{x}^*, \mathbf{y}')$ with the extra linear constraint that $h(S, j) \geq \frac{1}{2} d_j$. This is an ordinary multi-commodity flow network problem and can therefore be solved in polynomial time [8]. This can also be deduced from the *Matlab script* in Appendix B.

To construct a semi-integral solution, the residual demand d_j of each client must be saturated, whereas $\mathbf{f}$ currently only satisfies at least $\frac{1}{2} d_j$ by sending this demand to facilities in S. In order to ensure that the demand of each client is fully met, it is possible to double the opening of facilities in S and send up to 2 times the flow $\mathbf{f}$ to facilities in S. Doubling the opening y_i of facilities in S is possible, as these facilities were opened at most for a fraction of $\frac{1}{4}$, therefore, these facilities are opened at most for a fraction $\frac{1}{2}$. This is still allowed for a semi-integral solution (see Definition 5.2). The facilities in S are then opened twice as much. Therefore, also up to twice as much flow $h(i, j)$ can be sent over flow paths in $\mathbf{f}$. In order to ensure that each client gets exactly its demand of d_j saturated and no more, each client will get a fraction $\frac{h(i,j)}{h(S,j)} d_j$ of its demand from each facility in S. For example if a client j received $h(1, j) = 0.2 d_j$ from facility 1 and $h(2, j) = 0.6 d_j$ from facility 2, and nothing from other facilities, then in the semi-integral solution it will receive $\frac{h(1,j)}{h(S,j)} d_j = \frac{0.2}{0.2+0.6} d_j = 0.25 d_j$ from facility 1 and $\frac{0.6}{0.2+0.6} d_j = 0.75 d_j$ from facility 2. Because $h(S, j) \geq \frac{1}{2}$, the new connection between a client and facility $i \in S$ is no more than double the old connection. This is feasible, because the facility is also opened twice as much. Furthermore, each client's residual demand d_j is satisfied because $\sum_{i \in S} h(i, j) = h(S, j)$ by definition. Formally the semi-integral solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ becomes:

$$\hat{y}_i = \begin{cases} 1, & \text{if } i \in I; \\ 2y_i^*, & \text{if } i \in S; \end{cases}$$
$$\hat{x}_{ij} = \begin{cases} g_{ij}^*, & \text{if } i \in I, j \in \mathcal{D}; \\ \frac{h(i,j)}{h(S,j)} d_j, & \text{if } i \in S, j \in \mathcal{D}; \end{cases} \tag{41}$$

Summarizing, in this semi-integral solution, first facilities are fully opened with high fractional opening and the fractional solution assigns client to these facilities with g_{ij}^*. Then the residual demand for each facility is saturated by clients in S.

5.1.3 Cost of semi-integral solution

Finally it is shown that the cost of the semi-integral solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ described in Equation (41) is not more expensive than 8 times the fractional solution $(\mathbf{x}^*, \mathbf{y}^*)$.

First note that the opening of facilities in I costs no more than 4 times the fractional solution, as the facilities already had an opening of at least $y_i^* \geq 1/4$. The facilities in S are opened twice as much, $\hat{y}_i = 2y_i^*$, and therefore, the cost for opening these facilities is also twice as much. All together, the cost of opening facilities in the semi-integral solution costs no more than 4 times the opening cost of the fractional solution:

$$\sum_{i \in \mathcal{F}} o_i \hat{y}_i \leq \sum_{i \in I} 4 o_i y_i^* + \sum_{i \in S} 2 o_i y_i^* \leq 4 \sum_{i \in \mathcal{F}} o_i y_i^*.$$

For the connection cost of the semi-integral solution, the flow paths in $\mathbf{f}$ can be considered to find a bound for its cost. First consider the cost of the partial assignment g_{ij}^* to facilities in I. Partial assignment $\mathbf{g}^*$ was constructed from $\mathbf{z}$ by leaving out some of its edges, and $\mathbf{z}$ was the maximum b-matching between clients and facilities in I. The arcs in this matching had capacity of $2x_{ij}^*$, therefore, the cost of the assignment $\mathbf{z}$ is less than twice the total cost of the fractional assignment $\mathbf{x}^*$. Therefore the cost of the partial assignment $\mathbf{g}^*$ of the semi-integral solution is bounded by 2 times the cost of the fractional assignment:

$$\sum_{i \in I, j \in \mathcal{D}} c_{ij} g_{ij}^* \leq \sum_{i \in I, j \in \mathcal{D}} c_{ij} z_{ij} \leq 2 \sum_{i \in I, j \in \mathcal{D}} c_{ij} x_{ij}^* \leq 2 \sum_{i \in \mathcal{F}, j \in \mathcal{D}} c_{ij} x_{ij}^*$$

Finally the cost of the assignment of clients to fractionally opened facilities, S, are considered. The cost of the assignment of a client $j \in \mathcal{D}$ to some facility $i \in S$ happened on the base of these flow paths going from j to facility i, bringing $h(i,j)$ flow. The assignment of each client to a facility in S is $\frac{h(i,j)}{h(S,j)} d_j$. Due to Theorem 5.5, for each client $j \in \mathcal{D}$, at least half the residual demand d_j was sent to facilities in S by flow paths: $h(S,j) \geq d_j/2 \Rightarrow \frac{d_j}{h(S,j)} \geq 2$. The cost $c(P)$ of each flow paths $P \in \mathcal{P}_{ij}$ is greater than or equal to the direct connection cost c_{ij} due to the triangle inequality (see Equation (1)): $c(P) \leq c_{ij}, \forall P \in \mathcal{P}_{ij}$. Therefore, the connection cost to facilities in S is bounded by the cost of the flow paths:

$$\begin{aligned} \sum_{i \in S, j \in \mathcal{D}} c_{ij} \hat{x}_{ij} &= \sum_{i \in S, j \in \mathcal{D}} c_{ij} \tfrac{h(i,j)}{h(S,j)} d_j \\ &\leq \sum_{i \in S, j \in \mathcal{D}} c_{ij} 2h(i,j) \\ &\leq 2 \sum_{i \in S, j \in \mathcal{D}} \sum_{P \in \mathcal{P}_{ij}} c_{ij} f(P) \\ &\leq 2 \sum_{i \in S, j \in \mathcal{D}} \sum_{P \in \mathcal{P}_{ij}} c(P) f(P) \end{aligned}$$

Furthermore, the cost of all flow paths combined is bounded be the combined costs of all the arcs in $\mathbf{MFN}(\mathbf{g}^*, \mathbf{x}^*, \mathbf{y}')$ between nodes j^s and i, i.e. the arcs in the shaded area in Figure 16. This is because all flow path are part of $\mathbf{f}$, which is a feasible solution to $\mathbf{MFN}(\mathbf{g}^*, \mathbf{x}^*, \mathbf{y}')$. These arcs consist of arcs (j^s, i), which have capacity x^*_{ij}, and arcs (i, j^s), which have capacity g^*_{ij} (see Figure 16). It has just been shown that the total cost of the arcs g^*_{ij} was bounded by two times the cost of $\mathbf{x}$. Therefore, the total cost of all the flow paths can be bounded in terms of the cost of the fractional assignment $\mathbf{x}^*$:

$$\begin{aligned}\sum_{i\in S, j\in\mathcal{D}} c_{ij}\hat{x}_{ij} \leq 2 \sum_{i\in S, j\in\mathcal{D}} \sum_{P\in\mathcal{P}_{ij}} c(P)f(P) &\leq 2\left(\sum_{i\in\mathcal{F}, j\in\mathcal{D}} c_{ij}g^*_{ij} + \sum_{i\in\mathcal{F}, j\in\mathcal{D}} c_{ij}x^*_{ij}\right) \\ &\leq 2\left(\sum_{i\in\mathcal{F}, j\in\mathcal{D}} c_{ij}2x^*_{ij} + \sum_{i\in\mathcal{F}, j\in\mathcal{D}} c_{ij}x^*_{ij}\right) \\ &\leq 6\sum_{i\in\mathcal{F}, j\in\mathcal{D}} c_{ij}x^*_{ij}\end{aligned}$$

Finally the total cost of the semi-integral solution $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ is bounded by 8 times the total cost of the fractional solution $(\mathbf{x}^*, \mathbf{y}^*)$: $c(\hat{\mathbf{x}}, \hat{\mathbf{y}}) \leq 8 \cdot c(\mathbf{x}^*, \mathbf{y}^*)$:

$$\begin{aligned}c(\hat{\mathbf{x}}, \hat{\mathbf{y}}) &= \sum_{i\in\mathcal{F}} o_i y_i + \sum_{i\in\mathcal{F}, j\in\mathcal{D}} c_{ij}\hat{x}_{ij} \\ &= \sum_{i\in\mathcal{F}} o_i y_i + \sum_{i\in I, j\in\mathcal{D}} c_{ij}\hat{x}_{ij} + \sum_{i\in S, j\in\mathcal{D}} c_{ij}\hat{x}_{ij} \\ &\leq 4\sum_{i\in\mathcal{F}} o_i y^*_i + 2\sum_{i\in I, j\in\mathcal{D}} c_{ij}x^*_{ij} + 6\sum_{i\in\mathcal{F}, j\in\mathcal{D}} c_{ij}x^*_{ij} \\ &\leq 8\left(\sum_{i\in\mathcal{F}} o_i y^*_i + \sum_{i\in\mathcal{F}, j\in\mathcal{D}} c_{ij}x^*_{ij}\right) \\ &\leq 8 \cdot c(\mathbf{x}^*, \mathbf{y}^*)\end{aligned}$$

5.1.4 Existence of a half-saturating multi-commodity flow

In this section, Theorem 5.5 is further explained. As mentioned earlier, the full proof is omitted and only the idea behind the proof is given.

First Theorem 5.5 is restated:

Theorem 5.5. *There exist a feasible flow to* $\mathbf{MFN}(\mathbf{g}^*, \mathbf{x}^*, \mathbf{y}')$ *such that each client* j *sends at least half its residual demand* d_j *to facilities in* S*:* $h(S, j) \geq \frac{1}{2}d_j = \frac{1}{2}(1 - \sum_{i\in\mathcal{F}} g^*_{ij})$.

In the proof, first it is assumed that some valid flow does not saturate each half of the residual demand of each client. Then the client j with the lowest fraction saturated is considered. The proof states that it is possible to reroute flow, such that a new flow is possible from client j to facilities in S. This is possible, as long as not every client has at least half its demand saturated. Because the set of feasible flows is compact, this process cannot converge to some limit lower than $\frac{1}{2}d_j$. Therefore, a valid flow must exist such that each client is saturated for at least half its residual demand.

The rerouting of the flows such that an additional flow is possible will be explained further. In Figure 27, the rerouting process is illustrated. Let α be the lowest fraction of residual demand saturated by facilities in S, considering all clients with residual demand left:

$$\alpha = \min_{j:d_j>0} h(S,j)/d_j \tag{42}$$

Then let J be the set of clients that have exactly this fraction of their residual demand saturated. This is always at least one client, but this set can contain multiple clients when multiple clients all have the same, lowest fraction saturated.

$$J = \{j \in \mathcal{D} : h(S, j/d_j) = \alpha, d_j > 0\} \tag{43}$$

If every client has residual demand of $d_j = 0$, then there is nothing to prove. Assume that the set of clients with nonzero residual demand is not empty. Then assume that $\alpha < 1/2$, i.e. some client sends less than half its residual demand to facilities in S. It is known however that $\mathbf{MFN}(\mathbf{g}^*, \mathbf{x}^*, \mathbf{y}')$ has a feasible multi-commodity flow $\mathbf{f}$ that does satisfy all the residual demands. Therefore, the rest of the demand of client in J is sent to facilities in I. More specifically, the reachable facilities I_H were already saturated, and thus the rest of the demand is specifically sent to facilities in $I \setminus I_H$. The direct connection x^*_{ij} does not have the capacity, however, to send half the residual demand from J to $I \setminus I_H$ directly, and therefore, other flow paths also transport flow from J to $I \setminus I_H$. This means the flow passes through other clients, which are not in J, but in the set of reachable facilities $\mathcal{D}_H \setminus J$. Let k^s be the client at which such a path stops last before sinking the demand at a facility $i \in I \setminus I_H$, i.e. sending the demand over arc (i, i') and then to sink node j^t over (i', j^t).

It can be shown that k^s has nonzero residual demand [4], and because it is not in J, it sends more than fraction α of its demand to facilities in S. Summarizing, there is a flow path from a client $j \in J$ to $i \in I \setminus I_H$, which first meets client k^s before sinking at facility i. There are flow paths from client k^s to facilities in S such that more than a fraction α of its residual demand is sent to facilities in S. An example is shown in Figure 27. Now the flow paths originating from $j \in J$ and k^s are exchanged, that is, a small amount of flow ε from k^s is sent to $I \setminus I_H$ instead of to S, and the same amount of flow ε is sent from j to k^s to the facility in S via an old path of k^s. This way, the amount of flow sent from clients in $j \in J$ to facilities in S can always be increased so long as $\alpha < \frac{1}{2}$.

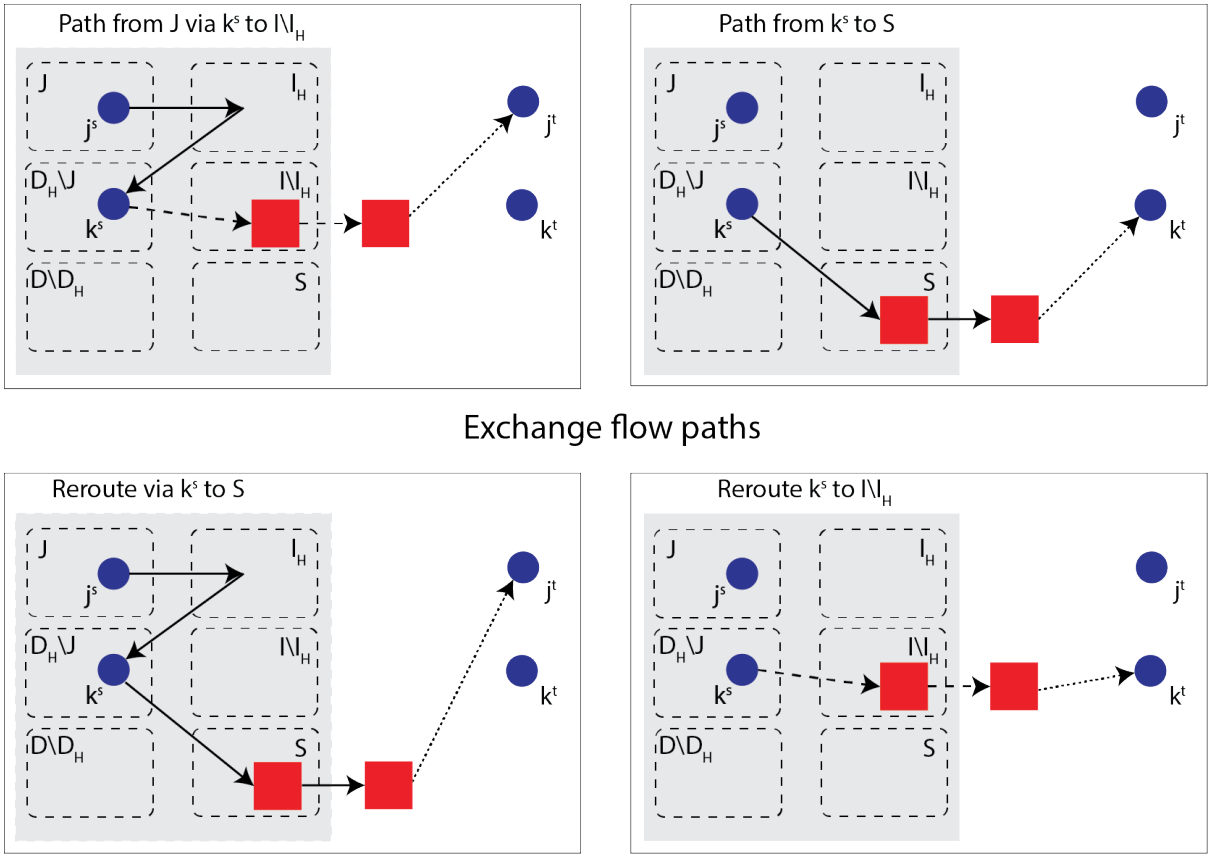


Figure 27: Illustration of the exchange of flow paths. Above are the initial flow paths. The solid line marks the path to be used to reroute a flow $j \in J$ to a facility in S, the dotted path will reroute an equal amount of flow from facility $k^s \in \mathcal{D}_H \setminus J$ to a facility $i \in I \setminus I_H$. This way, the same connections in the shaded area are still used.

5.2 From semi-integral solution to integral solution

In the last section a semi-integral solution, $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$, has been constructed. In this section, this semi-integral solution will be converted to an integral solution. The integral solution, $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$, will have cost no higher than 36 times the cost of the semi-integral solution: $c(\overline{\mathbf{x}}, \overline{\mathbf{y}}) \leq 36c(\hat{\mathbf{x}}, \hat{\mathbf{y}})$.

For constructing the integral solution, deciding what facilities to open is most important. Given a set of facilities to open, the cheapest assignment cost can be found in polynomial time [4]. The goal is to find a set of facilities to open such that the cost of opening is not too high and there also exists a cheap assignment to these facilities. This assignment is still allowed to be fractional. This is allowed because given a set of opened facilities, there is always an optimal solution that is integral. This is due to the fact that the polytope of possible assignments has integral vertices. The vertices of the polytope are integral, because each client has unit demand, and the facilities have integral capacity. Therefore, all intersections of constraints are integral.

The main problem is now to decide which facilities to open. The facilities in I have already been opened, so only the facilities in S are of interest. Now consider the assignment problem where all clients in $\mathcal{D}$ need to be assigned to facilities in S for exactly their residual demand d_j (after assignment to facilities in I). A facility $i \in S$ can only be assigned up to its residual capacity $U_i' = U_i - \sum_{i \in I} \hat{x}_{ij}$. The goal function is to minimize the cost of the assignment to residual facilities $i \in S$ plus the cost of opening the corresponding facilities. The linear relaxation of this problem is considered, so the facilities in S can be fractionally opened and the clients can be fractionally assigned. This problem will further on be referred to as $\mathrm{LP}_{\mathrm{demand}}$:

$$
\begin{aligned}
\text{minimize} \quad & \sum_{i \in S} o_i y_i \; + \sum_{i \in S, j \in \mathcal{D}} c_{ij} x_{ij} & & \\
\text{subject to} \quad & \sum_{i \in S} x_{ij} = \hat{d}_j & & \forall j \in \mathcal{D} \\
& \hat{d}_j y_i \geq x_{ij} & & \forall i \in S, j \in \mathcal{D} \\
& U_i' y_i \geq \sum_{j \in \mathcal{D}} x_{ij} & & \forall i \in S \\
& y_i \leq 1, & & \forall i \in S \\
& \mathbf{x}, \mathbf{y} \geq 0 & &
\end{aligned}
$$

Figure 28: $\mathrm{LP}_{\mathrm{demand}}$

Note that a solution to $\mathrm{LP}_{\mathrm{demand}}$ is given be taking the assignment variables to facilities in S, $\hat{\mathbf{x}}^{\mathbf{S}} = \{\hat{x}_{ij} : i \in S, j \in \mathcal{D}\}$, and the opening variables of facilities in S, $\hat{\mathbf{y}}^{\mathbf{S}} = \{\hat{y}_i, i \in S\}$. By leaving out the opening variable of facilities in I, $\hat{\mathbf{y}}^{\mathbf{I}} = \{\hat{y}_i : i \in I\}$, and assignments to facilities in I, $\hat{\mathbf{x}}^{\mathbf{I}} = \{\hat{x}_{ij}, i \in I, j \in \mathcal{D}\}$, a solution $(\hat{\mathbf{x}}^{\mathbf{S}}, \hat{\mathbf{y}}^{\mathbf{S}})$ is obtained for $\mathrm{LP}_{\mathrm{demand}}$. In this solution, all facilities in S are opened less than half $\hat{y}_i < \frac{1}{2}, i \in S$, as $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ is a semi-integral solution.

Abrams, Meyerson, Munagala and Plotkin proved it is possible to convert a fractional solution to $\mathrm{LP}_{\mathrm{demand}}$ to a solution with a fractional assignment, but an integral opening of facilities [9]. However, for this solution, the capacities of facilities must be increased and the new solution is also more expensive than the old one. Formally this is stated in Theorem 5.6:

Theorem 5.6. *Given a feasible, fractional solution* $(\mathbf{x}, \mathbf{y})$ *to* $\mathrm{LP}_{\mathrm{demand}}$*, a solution* $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ *can be constructed in polynomial time, such that* $\overline{\mathbf{y}}$ *is integral and satisfies* $\mathrm{LP}_{\mathrm{demand}}$*, if each capacity* U_i' *is replaced by* $2U_i'$*. The cost of the constructed solution is no more than 18 times the fractional solution,* $c(\overline{\mathbf{x}}, \overline{\mathbf{y}}) \leq 18c(\mathbf{x}, \mathbf{y})$*.*

The full proof of Theorem 5.6 is omitted in this thesis; it can be found in the work of Abrams, Meyerson, Managala and Plotkin [9]. The idea of the proof is to first find the fractional solution of $\text{LP}_{\text{demand}}$. Then each assignments x_{ij} is removed if it is too expensive. To compensate the loss in demand for client j, all capacities of all facilities are increased by a certain factor to allow the now unsaturated clients to be fully assigned. In this case, the capacity of the facilities are increased with a factor of 2. Then the algorithm will start selecting which facilities to open and which to close, looping as long as there are fractionally opened facilities left. At the start of each loop, the 'best' facility is found based on a search criterion considering the cost and capacity efficiency of each facility. This 'best' facility is fully opened and other clients, initially assigned to fractionally opened facilities, are reconnected to this facility so long as there is capacity left and reconnecting is not too expensive. If at a certain point some facility has no clients assigned to it at all, it is closed and removed from the set of feasible facilities. After the reconnection process, the 'best' facility is removed from the problem and so are all the assignments to this facility. The demand of each client is decreased by the amount it was assigned to the 'best' facility. This yields a new problem, where each client has smaller demand than before and with at least one possible facility less than the last problem. The new problem is solved linearly for a new solution and a new 'best' facility is chosen from the set of remaining facilities. The algorithm starts over as long as not facility is fully opened or fully closed. The output of the algorithm is a set of integrally opened facilities and a fractional assignment to these facilities. The capacity of each facility is doubled in the process.

Now consider the problem $\text{LP}_{\text{demand}}$ again with the set of clients $\mathcal{D}$, where each client has demand $\hat{d}_j = 1 - \sum_{i \in I} \hat{x}_{ij}$. The set of facilities is again S, however, the capacity of each facility is halved, $U'_i = \frac{1}{2}U_i, \forall i \in S$. This is done such that each capacity is returned to normal, after being doubled by the algorithm of Theorem 5.6. Note that with the new capacities U'_i, the semi-integral solution $(\hat{x}, \hat{y})$ may no longer be feasible, due to the fact that the capacity of each facility is halved and the constraint $U_i y_i \geq \sum_{j \in \mathcal{D}} x_{ij}$ may be violated now. This can easily be solved by doubling the opening $\hat{y}_i$ of each facility. This way, the mentioned constraint is no longer violated. Only the constraint $y_i \leq 1$ may be violated now. This is not the case, however, because $\hat{y}_i \leq \frac{1}{2}, \forall i \in S$ by definition of the semi-integral solution, and therefore, $2\hat{y}_i \leq 1$. Thus, $(\hat{\mathbf{x}}^{\mathbf{S}}, \mathbf{2}\hat{\mathbf{y}}^{\mathbf{S}})$ is feasible to the $\text{LP}_{\text{demand}}$ with facility capacities equal to $U'_i = \frac{1}{2}U_i, i \in S$. Note that doubling the opening of the facilities in S is at most twice as expensive as the original opening, therefore $c(\hat{\mathbf{x}}^{\mathbf{S}}, \mathbf{2}\hat{\mathbf{y}}^{\mathbf{S}}) \leq 2c(\hat{\mathbf{x}}^{\mathbf{S}}, \hat{\mathbf{y}}^{\mathbf{S}})$.

Now the algorithm of Theorem 5.6 is used and a solution $(\overline{\mathbf{x}}^{\mathbf{S}}, \overline{\mathbf{y}}^{\mathbf{S}})$ is obtained, which is a feasible solution to $\text{LP}_{\text{demand}}$ with doubled facility capacities. This solution costs at most 18 times the cost of $(\hat{\mathbf{x}}^{\mathbf{S}}, \mathbf{2}\hat{\mathbf{y}}^{\mathbf{S}})$: $c(\overline{\mathbf{x}}^{\mathbf{S}}, \overline{\mathbf{y}}^{\mathbf{S}}) \leq 18c(\hat{\mathbf{x}}^{\mathbf{S}}, \mathbf{2}\hat{\mathbf{y}}^{\mathbf{S}})$. The facility capacities were $U'_i = \frac{1}{2}U_i$, and therefore, the solution $(\overline{\mathbf{x}}^{\mathbf{S}}, \overline{\mathbf{y}}^{\mathbf{S}})$ is feasible for the facilities with capacities U_i. The pre-final solution $(\overline{\mathbf{x}}', \overline{\mathbf{y}}) \in [0,1]^{I \times \mathcal{D}} \times \{0,1\}^{\mathcal{F}}$ is now obtained by combining $(\hat{\mathbf{x}}^{\mathbf{I}}, \hat{\mathbf{y}}^{\mathbf{I}}) \in [0,1]^{I \times \mathcal{D}} \times \{1\}^{I}$ and $(\overline{\mathbf{x}}^{\mathbf{S}}, \overline{\mathbf{y}}^{\mathbf{S}}) \in [0,1]^{S \times \mathcal{D}} \times \{0,1\}^{S}$. In the pre-final solution, all facilities are either fully opened or closed. The assignment variables may still be fractional, however. Now the original CFL matching problem is reconsidered, except that the integral opening variables are known. Let F be the set of opened facilities, i.e. $F = \{i \in \mathcal{F} : \overline{y}_i = 1\}$. The formulation of the b-matching problem of clients to opened facilities F is shown in Figure 29.

At the start of this section it was claimed that there is always an optimal solution to this problem that is integral. Due to the fact that each client has unit demand and each facility $i \in F$ has integral capacity, the polytope corresponding to the constraints in the b-matching LP has integral vertices [4]. This can be interpreted as follows: if two facilities are both partially assigned to the same two facilities, then it is always cheaper (or with equal cost) to fully assign one client to one facility and the other client to the other facility. This can be extended to the

$$
\begin{aligned}
&\text{minimize} && \sum_{i\in F, j\in\mathcal{D}} c_{ij}x_{ij} && \\
&\text{subject to} && \sum_{i\in F} x_{ij} = 1, && \forall j \in \mathcal{D} \\
& && \sum_{j\in\mathcal{D}} x_{ij} \leq U_i, && \forall i \in F \\
& && \mathbf{x} \geq 0 &&
\end{aligned}
$$

Figure 29: b-matching LP

full set of clients and facilities. Figure 30 illustrates this principle.

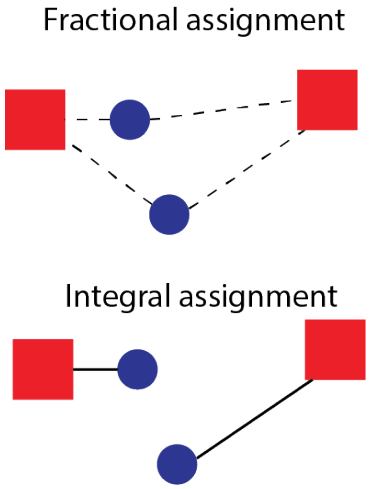


Figure 30: A fractional assignment and its cheaper integral variant. The distance between client and facility represents the cost. Although the lower client needs to travel further to the right facility, its extra cost is compensated, as the upper client can now send its full demand to the nearby, left facility.

It is known that the pre-final solution $(\overline{\mathbf{x}}', \overline{\mathbf{y}})$ is feasible for the b-matching of Figure 29, and therefore, a fully integral solution $(\overline{\mathbf{x}}, \overline{\mathbf{y}})$ exists with cost cheaper than the pre-final solution. This is the final, integral solution of the full algorithm.

Now an upper bound for the cost of the integral solution is shown.

$$
\begin{aligned}
c(\overline{\mathbf{x}}, \overline{\mathbf{y}}) &\leq c(\overline{\mathbf{x}}', \overline{\mathbf{y}}) \\
&= c(\hat{\mathbf{x}}^{\mathbf{I}}, \hat{\mathbf{y}}^{\mathbf{I}}) + c(\overline{\mathbf{x}}^{\mathbf{S}}, \overline{\mathbf{y}}^{\mathbf{S}}) \\
&\leq c(\hat{\mathbf{x}}^{\mathbf{I}}, \hat{\mathbf{y}}^{\mathbf{I}}) + 18c(\hat{\mathbf{x}}^{\mathbf{S}}, \mathbf{2}\hat{\mathbf{y}}^{\mathbf{S}}) \\
&\leq c(\hat{\mathbf{x}}^{\mathbf{I}}, \hat{\mathbf{y}}^{\mathbf{I}}) + 36c(\hat{\mathbf{x}}^{\mathbf{S}}, \hat{\mathbf{y}}^{\mathbf{S}}) \\
&\leq 36c(\hat{\mathbf{x}}^{\mathbf{I}}, \hat{\mathbf{y}}^{\mathbf{I}}) + 36c(\hat{\mathbf{x}}^{\mathbf{S}}, \hat{\mathbf{y}}^{\mathbf{S}}) \\
&= 36(\hat{\mathbf{x}}, \hat{\mathbf{y}})
\end{aligned}
$$

This proves that an integral solution to CFL exist with cost no more expensive than 36 times

the semi-integral solution. Recall that the semi-integral solution had cost at most 8 times the linear optimal solution of MFN-LP, which was a relaxation of CFL. Therefore, a feasible integral solution always exists, which has cost of at most $8 \cdot 36 = 288$ times the cost of the optimal, fractional solution to the relaxation of CFL. Therefore, the integrality gap of the relaxation MFN-LP is at most 288 and it is proven that there exists a formulation of CFL with bounded integrality gap.

6 Conclusions and recommendations

A relaxation of the capacitated facility location problem with a finite integrality gap of at most 288 has been presented. First a relaxation of the minimum knapsack problem was shown to have an integrality gap of at most 2, then some of the techniques used for the minimum knapsack problems were also used for the capacitated facility location problem. The algorithm proposed by An et al. [4] for the capacitated facility location problem has been illustrated and the LP-relaxation has been investigated.

For further strengthening the integrality gap of the CFL, it is recommended to first find a smaller, lower bound for the algorithm of An et al. The authors apply many excessive roundings in order to keep the algorithms as clear as possible. It is expected the upper bound of 288 can be greatly reduced by separating cases or by dividing the total cost in separate contributions and finding an upper bound for each contribution. Otherwise, a worst-case example may be constructed for which the bound of 288 is actually obtained by the algorithm of An et al. Finally, it is also recommended to find a different relaxation of the CFL with finite integrality gap.

References

[1] D.B. Shmoys, É. Tardos, and K. Aardal. Approximation algorithms for facility location problems. *ACM*, (29):265–274, 1997.

[2] A. Aggarwal, A. Louis, M. Bansal, N. Garg, N. Gupta, S. Gupta, and S. Jain. A 3-approximation algorithm for the facility location problem with uniform capacities. *Mathematical Programming*, 141(1-2):527–547, 2013.

[3] M. Bansal, N. Garg, and N. Gupta. A 5-approximation for capacitated facility location. In *European Symposium on Algorithms*, pages 133–144. Springer, 2012.

[4] H.C. An, M. Singh, and O. Svensson. Lp-based algorithms for capacitated facility location. *FOCS*, (55):256–265, 2014.

[5] R.D. Carr, L.K. Fleischer, V.J. Leung, and C.A. Phillips. Strengthening integrality gaps for capacitated network design and covering problems. *SODA*, pages 106–115, 2000.

[6] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, Inc., 1998.

[7] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer-Verlag, 2004.

[8] R.K. Ahuja, T.L. Magnanti, and J.B. Orlin. *Network flows, theory, algorithms, and applications*. Prentice-Hall, Inc., 1993.

[9] Z. Abrams, A. Meyerson, K. Munagala, and S. Plotkin. On the integrality gap of capacitated facility location. Technical report, Citeseer, 2002.

7 Appendix

7.1 Appendix A: Minimum knapsack code

```
%% Knapsack Problem
%The algorithm as proposed and proved by Carr, Fleisher, Leung and Phillips
%Script written by Pascal B.J. de Koster

    %System input
clc;
format short
    %Length of Y
n=5;
    %Minimize O*Y, Under Conditions U*Y<=D
    %Costs O, Demand D, Capacity U
O=[10; 3; 5; 8; 11];
D0=80;
U0=[39 11 32 43 31];
% O=[1; 10; 1];
% D0=100;
% U0=[11 90 12];

    %Lower and upper bound for Y
lb=zeros(n,1);
ub=ones(n,1);

    %Parts for Conditions Aeq Y=Beq, there are no equalities.
Aeq=[];
Beq=[];

    %Use SimpleY instead of floating point
%options = optimoptions('linprog','Algorithm','dual-simplex');

%% Simple Linear Knapsack Problem
    %In this part the ordinary Knapsack Problem is linearily solved without
    %any extra constraints. The solution vector Y can contain fractions.
[Y0,LinOpt0]=linprog(O,-U0,-D0,Aeq,Beq,lb,ub,[],options);

    %U divided by C, so Value per Cost
UdO0=U0'./O;

display(UdO0);
display(Y0);
display(LinOpt0);

%% Add additional constraints for the Strenghtened Linear Problem
    %In this part the Knapsack Problem is strenghtened with eYtra
    %constraints: the feasable solution must be feasible for all subsets A
    %of E with capacity U0A smaller than the demand D0.

    %U and D will be eYtended with eYtra conditions.
U=U0;
D=D0;

    %Check for all n-1 possible eYtra constraints of partial covers.
for i=1:2^n-1;
        %possible subset of edges A, denoted with 0's and 1's.
    A=decimalToBinaryVector(i,n,'LSBFirst')';
```

```
    U0A=U0*A;
    if (U0A<D0)
            %D_A is the residual demand D(A) after the value is subtracted
            %from UA from the demand D
        D_A=D0-U0A;
            %Convert each weight to U_A(e)=min{U(e),U(A)}
        U_A=min([U0; D_A.*ones(1,n)]);
        U_A=U_A.*(1-A');
        U=[U;U_A];
        D=[D;D_A];
    end
end

    %Solve the new new linear problem with eYtra constraints

[Y,LinOpt]=linprog(O,-U,-D,Aeq,Beq,lb,ub);

display(Y)
display(LinOpt)

%% Convert the linear solution to an integer solution

    %The elements with Y(e)>1/2 are rounded up and are part of the solution
Y_int=round(Y);
    %r is the smallest common multiple of the denominators of the elements
    %in Y smaller than 1/2. First the elements smaller than 1/2 are sorted:
Y_low=Y.*(Y<1/2);
    %Then the denominators are determined.
    %NOTA BENE: rat only approYimates the double, Matlab only stores
    %doubles, not rationals.
[Num,Den]=rat(Y_low);
display(Num);
display(Den);
    %Calculate the least common multiple of the denominators
r=double(lcm(sym(Den)));
display(r);
    %Create r buckets each with n components, indicating if each buckets
    %contains each item.
Buckets=zeros(n, r);
    %a(e) is the number each element e appears. a(e):=2rY(e)
a=2*r*Num./Den;
b=0;
for i=1:n
    for j=1:a(i)
        b=mod(b,r)+1;
        Buckets(i,b)=1;
    end
end

%% Find the bucket with the lowest cost

    %Find the unique Buckets and their costs
Buckets_Unique=unique(Buckets','rows')';
BucketCosts=O'*Buckets_Unique;
    %Find the cheapest Bucket
[BucketCost,i] = min(BucketCosts);
Bucket_Cheapest=Buckets_Unique(i);
display(BucketCost);
display(Bucket_Cheapest);
```

```
%% Find the final solution

    %The final solution vector is the initially rounded up vector plus the
    %vectors in the cheapest bucket
Y_Final=Y_int+Bucket_Cheapest;
Cost_Final=O'*Y_Final;

display(Cost_Final);
display(Y_Final);

    %compare results of Linear, stregthened linear and strenghtened integer
    %costs
format shortG
fprintf('Linear optimal cost       = %f \n', LinOpt0);
fprintf('Strenghtened linear cost  = %f \n', LinOpt);
fprintf('Strengthened integer cost = %i \n', Cost_Final);
```

7.2 Appendix B: Capacitated facility location code

```
%% Capacitated Facility Problem
%The algorithm as proposed and proved by An, Singh and Svensson
%Script written by Pascal B.J. de Koster

    %System input
clc
format short
    %n=number of Facilities |F|, m=number of Clients |D|
n=2;    %Facitilies
m=3;    %Clients
    %Minimize Sum_i(Oi*Yi)+Sum_ij(Cij*Xij),
    %Under Conditions: Sum_j(Xij)<=Ui, Sum_i(Xij)=1

    %Opening cost O, Connection cost C, Capacity U
if (n==1 || m ==1)
    display('Solution is trivial')
    return
end


    %Generate an area in which the facilities and clients are located
size=100;
    %Generate positions for the facilities
FacX=floor((size+1)*(rand(n,1)+rand(n,1))/2);
FacY=floor((size+1)*(rand(n,1)+rand(n,1))/2);
    %Generate positions for the clients
CliX=floor((size+1)*(rand(m,1)+rand(m,1))/2);
CliY=floor((size+1)*(rand(m,1)+rand(m,1))/2);
    %plot all locations
figure
scatter(FacX,FacY,'s','b')
hold on
scatter(CliX,CliY,'r')
legend('Facilities','Clients')
xlim([0,100]);
ylim([0,100]);
hold off

    %Generate facility capacities U and opening costs O
U=poissrnd(ceil(2*m/n),[n,1]);

%     %Insert manual data if desired
% U=[2;2];

O=10*poissrnd(round(size/10),[n,1]);
display(U);
display(O);

    %Generate supply costs as the distance between facility i and client j
    %using the Manhattan-metric d((x1,y1),(x2,y2))=|x2-x1|+|y2-y1|
C=zeros(n,m);
for i=1:n
    for j=1:m
        C(i,j)=abs(FacX(i)-CliX(j))+abs(FacY(i)-CliY(j));
    end
end
display(C)
```

```
%% Linear Capacitated Facility Location
    %In this part the CFL is linearily solved without
    %any extra constraints. The solution vector can contain fractions.

    %Total Cost vector o_1,...,o_n, c_11, c_21,..., c_n1,c_12,...c_nm
    %YX0= y_1,...,y_n,x_11,x_21,...,x_n1,x_12,...x_nm
 OC0=[O;C(:)];
    %Facility conditions: Sum_j(Xij)<=Ui*Yi
    %=> -Ui*Yi+Sum_j(Xij)<=0
U0=-diag(U);
for j=1:m
    U0=[U0,eye(n,n)];
end
LE0=zeros(n,1);
    %Client conditions: Sum_i(Xij)=1
D0=ones(1,n);
for i=2:m
    D0=blkdiag(D0,ones(1,n));
end
D0=[zeros(m,n),D0];
EE0=ones(m,1);

if (n<11 && m<11)
    display(OC0)
    display(U0);
    display(D0);
end

    %Use Simplex instead of floating point
options = optimoptions('linprog','Algorithm','dual-simplex');


    %Solve the linear problem
    %min OC0*YX0;
    %S.T.:
    %U0*YX0<=LE0;
    %D0*YX0=EE0
    %0<=YX0<=1
[YX0,LinOpt0]=linprog(OC0,U0,LE0,D0,EE0,...
    zeros(n+n*m,1),ones(n+n*m,1),[],options);

    %Linear Solution
Y0=YX0(1:n,1);
X0=YX0(n+1:n+n*m,1);
X0=vec2mat(X0,n)';

display(Y0)
display(X0)
display(LinOpt0)

figure
scatter(FacX,FacY,100.*Y0+10,'s','filled','b')
hold on
scatter(CliX,CliY,'r','filled')
legend('Facilities','Clients')
for i=1:n
    for j=1:m
        if (X0(i,j)>0)
            line([FacX(i),CliX(j)],[FacY(i),CliY(j)],'LineWidth',...
```

```
                X0(i,j),'Color','black')
        end
    end
end
xlim([0,100]);
ylim([0,100]);

b = num2str(rats(Y0)); c = cellstr(b);
dx=-3; dy=5;    %Displacement so the text does not overlay the data points
text(FacX+dx, FacY+dy, c);
hold off

%% Add additional constraints to the CFL
    %DO NOT RUN THIS CODE FOR N+M=>7! THIS WILL TAKE EXTREMELY LONG

%k=25; m=2; n=3; %ty jenny :)

    %Conditions for the strengthened, fractional (f) CFL.
OCf=OC0;
Uf=U0;
Df=D0;
EEf=EE0;
LEf=LE0;

fprintf('This run will take about %i steps, \n', ((n+1)^m*2^(2*n+2*m)));
fprintf('%i fractional solution x %i constraints \n',[((n+1)^m),...
    (2^(2*n+2*m))]);

for k=0:(n+1)^m-1
        %Counter
    if mod(k,1)==0
        fprintf('Feasible solution %i out of %i \n', k+1, (n+1)^m);
    end

        %Generation of all integer partial solutions. A is a vector with
        %where each element stands for the facility element j is send to.
        %Each of the n clients can be send to m possible facilities or
        %no facility at all, so m+1 possible choices.
        %Therefore, there are (m+1)^n different
        %partial solution, not all of which are feasible though.
    A=[];
        %Connection matrix of the partial solution g
    g=zeros(n,m);
    k2=k;
    for j=1:m;
        A=[A, mod(k2,(n+1))];
        k2=(k2-A(j))/(n+1);
        if A(j)>0;
            g(A(j),j)=1;
        end
    end

        %Checking the feasiblility of each integral partial solution
    if (g*ones(m,1)<=U)
            %The residual demands of each client j with partial solution g
        dj=ones(1,m)-ones(1,n)*g;
            %Creating the full connection graph consisting of a start node
            %s and an end node t, the nodes js, i, i' and jt.
        MFN=zeros(2*(n+m+1),2*(n+m+1));
            %We split this matrix in several separate sections
```

```
    %First, the edges from start j to clients js are considered
s_js=dj';
    %Second, the edges from js to i
js_i=ones(n,m);
    %Third, the edges from i to js
i_js=g';
    %Fourth, the edges from i to i'
i_ii=diag(U-g*ones(m,1));
    %Fifth, the edges from i' to jt
ii_jt=dj'*ones(1,n);
    %Finally, the edges for jt to the end node t
jt_t=dj;

MFN=[zeros(1,2*n+2*m+2);...
    s_js,zeros(m,m),i_js,zeros(m,n+m+1);...
    zeros(n,1),js_i,zeros(n,2*n+m+1);...
    zeros(n,m+1),i_ii,zeros(n,n+m+1);...
    zeros(m,n+m+1),ii_jt,zeros(m,m+1);...
    zeros(1,m+2*n+1),jt_t,0];

%       %Making a plot of each situation
%   figure
%   scatter([0,ones(1,m),2*ones(1,n),3*ones(1,n),4*ones(1,m),5],...
%       [0,(m-1:-2:-m+1)/m,(n-1:-2:-n+1)/n,(n-1:-2:-n+1)/n,...
%       (m-1:-2:-m+1)/m,0],'filled')
%   hold on
%   for j=1:m
%       if dj(j)>0
%           line([0,1],[0,(m+1-j*2)/m])
%           line([4,5],[(m+1-j*2)/m,0])
%       end
%       for i=1:n
%           line([1,2],[(m+1-j*2)/m,(n+1-i*2)/n])
%           line([3,4],[(n+1-i*2)/n,(m+1-j*2)/m])
%       end
%   end
%   for i=1:n
%       line([2,3],[(n+1-i*2)/n,(n+1-i*2)/n])
%   end
%   hold off

    %Checking all possible cutsets
for s=0:2^(2*m+2*n)-1
%           %Counter 2
%       if mod(s,1000)==0
%          fprintf('Constraint %i out of %i \n', s, 2^(2*m+2*n));
%       end
        %Set stands for the nodes which will be included in each
        %cutset
    Set=decimalToBinaryVector(s,2*m+2*n,'LSBFirst');
    Set=[1,Set,0];
    CutSet=(1-Set)'*Set;
    MFN2=CutSet.*MFN;
        %Adding the conditions for Yi
    U1=[];
    for i=1:n
        U1=[U1,MFN2(1+m+n+i,1+m+i)+...
            sum(MFN2(1+m+2*n+(1:m),1+m+n+i))];
    end
        %Adding the conditions for Xij
```

```
            for j=1:m
                for i=1:n
                    U1=[U1,MFN2(1+m+i,1+j)];
                end
            end
            Uf=[Uf;-U1];
                %Each cutset must be greater or equal the total residual
                %demand. We can separate two subsets, MFN2 includes s and
                %excludes t. We consider all arcs going from MFN2 to
                %MFN\MFN2 and impose that the sum of capacities of all arcs
                %must be greater than or equal to the total demanded
                %capacity
            LE1=-(sum(dj)-sum(sum(MFN2(1:1+m,:)))...
                -sum(MFN2(2*(m+n+1),:)));
            LEf=[LEf;LE1];
        end
    end
end

%% Calculate the solution to the strenghtened CFL
    %All constraints are added, now solve the new problem

display('Computing optimum...')

[YXf,LinOptf]=linprog(OCf,Uf,LEf,Df,EEf,...
    zeros(n+n*m,1),ones(n+n*m,1),[],options);

Yf=YXf(1:n,1);
Xf=YXf(n+1:n+n*m,1);
Xf=vec2mat(Xf,n)';

display(Yf)
display(Xf)
display(LinOptf)
fprintf('Linear optimal cost           = %f \n', LinOpt0);
fprintf('Strenghtened fractional cost  = %f \n', LinOptf);

figure
scatter(FacX,FacY,100.*Y0+10,'s','filled','b')
hold on
title('Strengthened, fractional');
scatter(CliX,CliY,'r','filled')
legend('Facilities','Clients')
for i=1:n
    for j=1:m
        if (Xf(i,j)>0)
            line([FacX(i),CliX(j)],[FacY(i),CliY(j)],'LineWidth',...
                Xf(i,j),'Color','black')
        end
    end
end
xlim([0,100]);
ylim([0,100]);

b = num2str(rats(Yf)); c = cellstr(b);
dx=-3; dy=5;    %Displacement so the text does not overlay the data points
text(FacX+dx, FacY+dy, c);
hold off

%% From fractional to semi-integral solution
```

```
    %The algortihm as described by An, Singh and Svensson
    %Semi integral solution is marked as Ys, Xs, YXs
Ys=zeros(n,1);
    %Integral opened facilities
I=zeros(n,1);
for i=1:n
    if Yf(i)>=1/4
        Ys(i)=1;
        I(i)=1;
    else
        Ys(i)=Yf(i);
    end
end
S=ones(n,1)-I;

    %G is a bibartite graph given by the clients D and the integer opened
    %facilities I. There are arcs of capacity 2*X_ij between client j and
    %facility i.
X2=2*(I*ones(1,m).*Xf);
U_int=(U.*I);

G2=[zeros(1,1+m+n+1);...
    ones(m,1),zeros(m,m+n+1);...
    zeros(n,1),X2,zeros(n,n+1);...
    zeros(1,1+m),U_int',0];

    %The vector with the indices of opened facilities
I2=find(I);
[~,~,U_int2]=find(U_int);
X2(all(X2==0,2),:)=[];

names={'s'};
for j=1:m
    names(1,j+1)={strcat('D',num2str(j))};
end
for i=1:n
    names(1,1+m+i)={strcat('F',num2str(i))};
end
names(1,1+m+n+1)={'t'};
display(names)
num2str(j)
num2str(i)
G=digraph(G2',names);
figure
plot(G,'EdgeLabel',G.Edges.Weight);

[mf,GF]=maxflow(G,1,m+n+2,'augmentpath');
display(mf);
figure
H=plot(G,'EdgeLabel',G.Edges.Weight);
H.EdgeLabel = {};
highlight(H,GF,'EdgeColor','r','LineWidth',2);
st = GF.Edges.EndNodes;
labeledge(H,st(:,1),st(:,2),GF.Edges.Weight);

%% Create residual network H

    %Z is the (n-by-m) matrix from the flow from facilities to clients.
    %It is the solution to the max flow in the last paragraph.
Z=zeros(m+n,m+n);
```

```
for k=1:length(st)
    if (st(k,1)~=1 && st(k,2)~=1+m+n+1)
        Z(st(k,1)-1,st(k,2)-1)=GF.Edges.Weight(k);
    end
end
Z=Z';
display(Z);

    %H is the boolean residual flow network of the bipartite graph G.
    %A node (j,i) is included if Z_ij<2X_ij and (i,j) is included if Z_ij>0
H=zeros(m+n,m+n);
for i=1:n
    for j=1:m
        if Z(m+i,j)<2*Xf(i,j)-10^-14
            H(m+i,j)=1;
        end
        if Z(m+i,j)>0
            H(j,m+i)=1;
        end
    end
end

names_H={};
for j=1:m
    names_H(1,j)={strcat('D',num2str(j))};
end
for i=1:n
    names_H(1,m+i)={strcat('F',num2str(i))};
end

H_digraph=digraph(H,names_H);
XData=[-1*ones(1,m),ones(1,n)];
YData=[((m-1):-2:(-m+1))./m,((n-1):-2:(-n+1))./n];
figure
plot(H_digraph,'XData',XData,'YData',YData,'EdgeLabel',H_digraph.Edges.Weight);

    %Set of clients saturated by Z
D_sat=zeros(m,1);
for j=1:m
    if sum(Z(m+1:m+n,j))>=1-10^(-14)
        D_sat(j,1)=1;
    end
end

    %Set of unsaturated clients
D_unsat=ones(m,1)-D_sat;
    %Set of clients and facilities reachable by an unsaturated client,
    %Initially this is only the set of unsaturated clients itself
DI_reachable=[D_unsat;zeros(n,1)];
DI_reachable2=zeros(m+n,1);
    %Iterations are made such that a facility or client is added to the set
    %of reachables if they are reachable by a client or facility in the
    %previous reachable set. The algorithm stops when nothing is added to
    %the set of reachables anymore.
while sum(~(DI_reachable==DI_reachable2))>=1;
    DI_reachable2=sign((DI_reachable'*H)');
    DI_reachable=sign(DI_reachable+DI_reachable2);
end
display(DI_reachable)
```

```
I_reachable=DI_reachable(m+1:m+n,1);
D_reachable=DI_reachable(1:m,1);

g=zeros(n,m);
for i=1:n
    for j=1:m
        if I_reachable(i)==1
            g(i,j)=Z(m+i,j);
        end
        if I_reachable(i)~=1 && D_reachable(j)~=1
            g(i,j)=Z(m+i,j);
        end
    end
end

%% Construct the final semi-integral solution (Xhat,Yhat)

%NA is the Number of Arcs in MFN(g*,x*,y*)
NA=(2*(n*m)+(n)+(n*m));
    %fG = goalfunction, only consists of zeros, because we are only
    %interested in the flow feasible to MFN(g*,x*,y*)
fF=zeros(m*NA,1);
    %AF*F<=bF
AF=zeros(NA+m,m*NA);
bF=zeros(NA+m,1);

    %From js to i
for s=1:m*n
    for k=0:m-1
        AF(s,k*NA+s)=1;
    end
end
    %From i to js
for s=m*n+(1:m*n)
    for k=0:m-1
        AF(s,k*NA+s)=1;
    end
end
    %From i to i'
for s=2*(m*n)+(1:n)
    for k=0:m-1
        AF(s,k*NA+s)=1;
    end
end
    %From i'to jt
for s=2*(m*n)+n+(1:m*n)
    for k=0:m-1
        AF(s,k*NA+s)=1;
    end
end
    %For bF, Xf,Yf,g and dj are required.
djF=ones(1,m)-ones(1,n)*g;
    %For machine precision
for j=1:m
    if abs(djF(1,j))<10^(-14)
        djF(1,j)=0;
    end
end

Cap_i_ii=Yf.*(U-g*ones(m,1));
```

```
Cap_ii_jt=Yf*djF;
bF=[Xf(:);g(:);Cap_i_ii(:);Cap_ii_jt(:)];
for s=1:length(bF)
    if abs(bF(s,1))<10^(-14)
        bF(s,1)=0;
    end
end

%Each client must dump at least half its residual demand at facilities
    %in S: h(S,j)>=dj
for k=0:m-1
    AF(NA+1+k,k*NA+2*n*m+(1:n))=-1.*S';
    bF(NA+1+k,1)=-1/2.*dj(k+1);
end

    %Now the constraints are added that all incoming flow must come out for
    %each node, in each commodity specific flow. There are 2m+2n nodes and
    %m commodities.

    %Number of arcs
Arcs=2*m+2*n;
AeqF=zeros(m*(2*m+2*n),m*NA);
beqF=zeros(m*(2*m+2*n),1);
for k=0:m-1
        %Starting nodes js, xij are going out
    for s=1:m
        AeqF(k*Arcs+s,k*NA+(s-1)*n+(1:n))=-ones(1,n);        %Ougoing x
        AeqF(k*Arcs+s,k*NA+m*n+(s-1)*n+(1:n))=1*ones(1,n);   %Incoming g
    end
    if k+1==s
        beqF(k*Arcs+s,1)=-dj(s);
    end
        %Nodes i (arcs g and i-i' are going out, x are going in)
    for s=1:n
        AeqF(k*Arcs+m+s,k*NA+(s:n:m*n))=ones(1,m);           %Incoming x
        AeqF(k*Arcs+m+s,k*NA+m*n+(s:n:m*n))=-1*ones(1,m);    %Outgoing g
        AeqF(k*Arcs+m+s,k*NA+2*m*n+s)=-1;                    %Outgoing i-i'
    end

        %Node i' arcs i-i' are going in, yd are going out
    for s=1:n
        AeqF(k*Arcs+m+n+s,k*NA+2*m*n+s)=1;                   %Incoming i-i'
        AeqF(k*Arcs+m+n+s,k*NA+2*m*n+n+(s:n:m*n))=-1*ones(1,m);
                                                            %Outgoing i'-jt
    end

        %End nodes jt, arcs yd are going in
    for s=1:m
        AeqF(k*Arcs+m+2*n+s,k*NA+2*m*n+n+(s-1)*n+(1:n))=ones(1,n);
                                                            %Incoming i'-jt
        if k+1==s
            beqF(k*Arcs+m+2*n+s,1)=dj(s);                       %Ougoing x
        end
    end
end

lbF=zeros(m*NA,1);
ubF=[];
for j=1:m
    ubF=[ubF;bF];
```

```
end

F=linprog(fF,AF,bF,AeqF,beqF,lbF,ubF);

for s=1:length(F)
    if abs(F(s,1))<10^(-3)
        F(s,1)=0;
    end
end

    %h(i,j) is the amount of flow from client j that uses the arc i-i'
h=zeros(n,m);
    %hS(j) is the amount of flow client j send to facilities in S
hS=zeros(m,1);
S=ones(n,1)-I;
for j=1:m
    for i=1:n
        h(i,j)=F((j-1)*NA+2*m*n+i,1);
    end
end
hS=S'*h;


    %Finally we constructe the semi-integral solution
Yhat=I+Yf.*S;

Xhat=g;
for j=1:m
    for i=1:n
        if S(i)==1 && hS(j)>1
            Xhat(i,j)=h(i,j)/hS(j);
        end
    end
end

YXhat=[Yhat(:);Xhat(:)];
display(Xhat)
display(Yhat)

figure
scatter(FacX,FacY,100.*Yhat+10,'s','filled','b')
hold on
scatter(CliX,CliY,'r','filled')
legend('Facilities','Clients')
for i=1:n
    for j=1:m
        if (X0(i,j)>0)
            line([FacX(i),CliX(j)],[FacY(i),CliY(j)],'LineWidth',...
                Xhat(i,j),'Color','black')
        end
    end
end
title('Semi-integral solution')
xlim([0,100]);
ylim([0,100]);

%% Conversion of semi-integral solution to an integral solution

    %No time was left for implementing this algorithm in Matlab
    %For the full algorithm, please see the work of Abrams, Meyerson,
```

```
    %Munagala, Plotkin in addition to the thesis.

%     %Sfrac contains all fractional openings, i.e. 0<yi<1;
% Sfrac=Yhat(Yhat>0 & Yhat<1);
%     %Nofrac is the number of fractional opened facilities
% Nofrac=length(Sfrac);
%
%     %Temporaral fractional opened facilities
% Y2=S;
%     %Temporaral fractional assignments
% X2=Xhat.*(S*ones(1,m));
%     %Double openings of facilities in S
% U2=U*Yhat*2*S;
%
% while Nofrac~=0
%     D=sum((Y2*ones(1,m)).*C.*Xhat);
%     AVG=(sum(C'.*(D'*ones(1,n)))./sum(C'))';
%     for i=1:n
%         SumD=0;
%         for j=1:m
%             if (C(i,j)<=4/3*D(j) && D(j)<=8/3*AVG(i))
%                 SumD=SumD+2;
%             end
%         end
%         U2(i)=min(U2(i),1);
%     end
% end
```