The metric uncapacitated facility location problem

The problem and the approximation algorithms

Amanda Fors
Abstract

The main purpose of this work is to present the basics and history of the metric uncapacitated facility location problem and give an introduction to the approximation algorithm of Vazirani, that gives an approximation guarantee of 3 for the optimal solution for the problem. Furthermore, the algorithm of Vazirani is demonstrated by an easy problem that shows the technicalities of the algorithm.

Sammanfattning

Syftet med denna rapport är att redogöra för grunderna och historien om det metriska okapaciterade anläggningsproblemet samt att ge en introduktion till approximationsalgoritmen av Vazirani, som ger en approximationsgaranti på 3 för det optimala värdet för problemet. Rapporten syftar även till att ge en analys av Vazirani’s algoritm med hjälp av ett simpelt problem som visar den teknik som algoritmen använder.
Acknowledgement

I would like to express a sincerest thanks to my supervisor, Per-Håkan Lundow, for the inspiration and support he has given me during my writing. He has always been very generous helping me with my questions.
1. Introduction

Since the beginning of the 1960’s the facility location problem has taken a central place in operation research and has many different applications in today’s society. The main purpose of the problem is to decide where to place a number of facilities, in order to cover the need of service for a number of cities, in the most efficient way. The most usual form of facilities is warehouses, hospitals, police stations and factories that produces some kind of services or products that the identified set of cities or customers are requiring.

The problem can appears in several forms, as for example uncapacitated or capacitated, discrete or continuous, metric or non-metric. These different types of facility location problems are discussed in the first section, with a focus on the metric uncapacitated facility location problem (metric UFL problem), the type of problem that the rest of the work is based on. In the first section we also introduce the binary integer problem formulation (BIP-formulation), an often used formulation of the metric UFL problem.

Because the metric UFL problem is NP-hard, several techniques have been developed for finding an approximated optimal solution for the problem. The linear program relaxation (LP-relaxation) of the BIP-formulation has a central role in these approximation algorithms because it makes the problem solvable in polynomial time. Solving optimum of the LP-relaxation gives a feasible approximated optimal solution to the metric UFL problem, as long as all variables $x_{ij}$ and $y_i$ have binary values. This is explained further in section 3, where the most common techniques for finding the optimal solution of the LP-relaxation are presented.

The approximation algorithms that are used for finding the approximated optimal solution for the metric UFL problem are usually constant factor approximation algorithms. These algorithms find a constant factor $\rho$, telling that the cost of the computed solution does not exceed $\rho$ times the optimal cost for any instance. In other words, $\rho$ is an approximation guarantee telling the ratio between the approximated solution and the optimal solution. This means, the closer the approximation factor $\rho$ is to value 1 the more exact the approximation algorithm is.

Actually, these constant factor approximation algorithms give a quite high approximation factor $\rho$ for many of these approximation algorithms. By Sviridenko it is proved that the best approximation guarantee that can be obtained by an approximation algorithm is $\rho = 1.463$, unless $P = NP$, see Ref. [2].

Section 4 contains a presentation and an analysis of a well known 3-approximation algorithm with $\rho = 3$ by Vazirani from 2001 [7], using LP-relaxation and linear programming duality (LP-duality), more precise a Primal-dual scheme. This algorithm gives an approximation factor at 3 in worst case and is often mentioned in other sources and articles when referring to a Primal-dual schema based approximation algorithm for the metric UFL problem.

Finally, the main purpose of this work is to present the basics and history of the metric uncapacitated facility location problem and give an introduction to the approximation algorithm of Vazirani, that gives an approximation guarantee of 3 for the optimal solution for the problem. Furthermore, the algorithm of Vazirani is demonstrated by an easy problem that shows the technicalities of the algorithm.
1.1 Notation list

\[ C \text{ the set of demand points (cities)} \]
\[ F \text{ the set of possible facility locations} \]

\[ f \in R_+^{\mid F \mid} \quad f \text{ belongs to all positive real numbers in dimension } \mid F \mid , \text{ where } \mid F \mid \text{ is the cardinality value of the set of possible facility locations} \]

\[ c \in R_+^{\mid C \mid} \quad c \text{ belongs to all positive real numbers in dimension } \mid C \mid , \text{ where } \mid C \mid \text{ is the cardinality value of the set of demand points (cities)} \]

\[ \phi(j) = i \quad \text{the assignment function, city } j \text{ is connected to facility } i \]

\[ c_{ij} \quad \text{service or connection cost for assigning city } j \text{ to facility } i \]
\[ f_i \quad \text{opening cost for facility } i \]

\[ x_{ij} \quad \text{indicator variable telling if city } j \text{ is connected to facility } i \]
\[ y_i \quad \text{indicator variable telling if facility } i \text{ is opened} \]

\[ \alpha_j \quad \text{the total price paid by city } j \]
\[ \beta_{ij} \quad \text{the price paid by city } j \text{ towards opening facility } i \]

\[ \maximize_x \quad \text{indicates: find maximum of the objective function, w.r.t the independent variable } x, \text{ in an optimization problem} \]

\[ \minimize_x \quad \text{indicates: find minimum of the objective function, w.r.t the independent variable } x, \text{ in an optimization problem} \]

\[ \text{subject to } \quad \text{specifies constraints of an optimization problem} \]

\[ \text{maximal independent subset} \quad \text{In graph theory; a maximal subset with vertices from a graph, where no pairs of vertices in this subset has an edge connecting them in the graph. Thus, every edge in the graph has maximum one vertex in the subset.} \]
2. Preliminaries

In this chapter a first presentation of the facility location problem with its different variations is given. Afterwards, there is a section describing the metric uncapacitated facility location problem (the metric UFL), which is a well-known type of the facility location problem. The chapter will then continue with focusing on the metric uncapacitated facility location problem and the complexity and approximability of the problem. Section 2 are mainly based on the work of Bumb and Vazirani [1,7].

2.1 The facility location problem

The main purpose of the facility location problem is to decide where to place a number of facilities, in order to cover the need of service for a number of cities, in the most efficient way. The most usual form of facilities is warehouses, hospitals, police stations and factories that produces some kind of services or products that the identified set of cities or customers are requiring. For example, it can be a retail company that wants to place its stores at the market on the most efficient way in relation to their customers. Also, it can be a city or municipality that must decide where to place their fire stations in order to cover the need of service for the residents. Today, the problem also has become important when it comes to placements of servers or data objects in communication networks to ensure the latency of access is optimized. However, the purpose of the problem is always the same, the interest of maximize the profit or to minimize the costs of the placement of the different facilities.

Other common elements for every facility location problem is that they always include a set of possible locations for facilities to be built on, or to be opened at for already built facilities, and information about the cost of building or opening a facility on that location. They also include a set of customers or demand points, with information about the demand that has to be covered and the cost of connecting the demand point to a certain facility. Further, the facility location problem is based on several conditions that have to be satisfied and an objective function that has to be optimized. The objective function is based on the cost of assigning the demand points to the facilities and the opening costs for these facilities. Notice, this is when there is a cost associated with connecting demand points to facilities. In other facility location problems a profit can be associated to the assignment and then the function of profit will be optimized.

Based on these common elements it is possible to divide the facility location problem into different types. For instance the division depends on whether the problem can be expressed with a finite set of facilities and demand points or with an infinite set. The problem is then called discrete respectively continuous. The partition can also be done based on whether the data is exact or based on probability. For example some parameter values can be given by probability distributions. This determines whether the problem is called deterministic or stochastic. Furthermore, a problem can be dynamic or static depending on if the time for establishing the location of a facility is important (dynamic), not only the location of the facility (static). At last an important classification of the facility location problem is based on whether the facilities are uncapacitated or capacitated in terms of how many demand points they can serve.

In this work we will study the metric uncapacitated facility locations problem that is static, deterministic and discrete. The metric property will be described in next section. As we will describe later, this kind of facility location problem is very hard to solve even though it is simple to formulate.
2.2 The metric uncapacitated facility location problem

The formulation of the metric uncapacitated facility location problem (metric UFL) contains a bipartite graph with bipartition \((F, C)\) where \(F\) is the set of potential locations for facilities \(i\) and \(C\) is the set of demand points \(j\), defined as cities. The cost of opening a facility at location \(i \in F\) is \(f_i\) and the cost of connecting a city \(j \in C\) to a facility \(i\) (i.e. activating edge \((i, j)\) in the bipartite graph) is \(c_{ij}\). This connection cost can also be named as a service cost because essentially \(c_{ij}\) is the cost of serving city \(j\) its demand from facility \(i\). Notice that \(f \in \mathbb{R}_+^{|F|}\) and \(c \in \mathbb{R}_+^{|F||C|}\). In this kind of formulation the demand \(d_j\) are assumed to be one unit for every city \(j\). This because it makes it easy to extend to arbitrary demands. This means the connection cost \(c_{ij}\) describe the cost to let facility \(i\) serve city \(j\) with one unit. A simple metric UFL problem with two potential facilities \(i\) and three cities \(j\) can be formulated as the bipartite graph in Fig. 2.1 below.

Further, the metric UFL problem are usually formulated as a binary integer program (BIP) with indicator variables \(x_{ij}\) and \(y_i\) defined as binary variables with values \(\{0, 1\}\), indicating whether city \(j\) is connected (served) to facility \(i\) respectively whether facility \(i\) is opened or not. The goal with solving the problem is to find a subset of facilities to be opened and to find an optimal assignment of cities to these opened facilities, all to a minimum total cost. For this, we have an assignment function \(\phi(j) = i\), telling that city \(j\) is assigned (connected) to facility \(i\) if \(\phi(j) = i\). The total cost is formulated in the objective function as the sum of every connection cost caused when connecting city \(j\) to facility \(i\), and the sum of opening costs for opened facilities. The binary integer program is formulated as,
The distinguish property for the metric UFL is that the connection costs \( c_{ij} \) satisfy the triangle inequality,

\[
c_{ij} \leq c_{ij} + c_{i'j} + c_{i'j'} \quad \text{for all } i, i' \in F \text{ and } j, j' \in C
\]

Thus, this means it is always most efficient to choose connection cost \( c_{ij} \) instead of \( c_{i'j} + c_{i'j'} + c_{ij} \) if we want city \( j \) to be served of facility \( i \). In many practical problems of the facility location problem, the connection costs corresponds to geometric distances, or to travel times, and therefore they are metric.

From the BIP-formulation (2.1) we can see that for the metric UFL problem every city \( j \) has to be assigned to at least one facility \( j \), i.e. this is what the first set of constraints say. Actually, it is allowed to connect one city to more than one facility but this is not an optimal scenario. Since the problem is uncapacitated the facilities will be able to serve every city that want to be served.

The second set of formulated constraints in (2.1) make sure that every facility that any cities are connected to is opened. A facility is declared as opened only if the cost of opening a facility, \( f_i \), is fully paid for. This means it is possible for a city to be denied service from a facility that offers the best service cost, if there are not enough with other cities that are interested in being served from the same facility.

In next section we will discuss the complexity of the problem and explain why it is a hard solvable problem. It also contains an introduction to the constant approximation algorithm as a way to find an approximated solution for the metric UFL problem.

### 2.3 Complexity and approximability

As mentioned before the metric UFL problem belongs to the class of hard types of optimization problems. Although the metric UFL problem is an easier type of the problem, e.g. comparing with the capacitated facility location problem, it is enough hard to be called a hard optimization problem. The fact that it is a hard solvable problem means we can not find an algorithm that solves the problem in polynomial time as for easy solvable problems, at least such an algorithm is not yet known. Also, it is very unlikely it exists.

The reason why the problem is hard to solve is because there are \( 2^{|F|} \) possible choices of facilities to go through, when deciding which facilities to open in order to obtain the optimal solution. This means we have to go through all possible subsets of facilities to be able to choose the best facilities to open and this makes the problem very hard to solve.
Though, once we have decided which facilities to be open we just connect each city \( j \) to the facility \( i \) with the smallest \( c_{ij} \), but it is hard to reach this point.

Using a greedy method trying to solve the problem will therefore be very expensive, for example the greedy algorithm proposed by Wolsey [10]. This algorithm describes a typical approach for a greedy method that starts with an initial possible solution (a subset of facilities) and successively goes through all neighbouring possible solutions, trying to find a better solution. The neighbouring solutions mean the subsets of facilities that arise from adding or removing a single element (facility) from the current subset. If a neighbouring solution is better than the current subset, the same procedure will start over again, trying to find an even better solution. When the algorithm no longer finds a better solution the algorithm will stop, and has then found a \textit{locally} optimal solution. For more details see Ref. [10].

The complexity class for a hard problem is \( NP \) or \( NPC \) and \( P \) for easy problems. Thus, \( P \) consists of problems with polynomial time algorithms, such as the UFL problem with no opening costs. It is established that the metric UFL problem is a \( NP \)-hard optimization problem. We also know it is widely assumed that \( P \subseteq NP \) and that \( P \neq NP \), since it would be very unlikely if \( P = NP \). Therefore, unless \( P = NP \), we know that there is no algorithm for a problem in \( NP \) that solves the problem \textit{exactly} in polynomial time. The best alternative way trying to solve the problem is then to find an algorithm that approximates the solution of the problem. We have to conciliate that no exact solution will be found.

One way to approximate the solution of the problem is to use a constant approximation algorithm, also called \( \rho \)-approximation algorithm, which gives the constant factor \( \rho \in R \). The constant factor \( \rho \) tells that the cost of the solution computed with the approximation algorithm, does not exceed \( \rho \) times the real optimal cost for any instance. In other words, \( \rho \) is an approximation guarantee for the approximation algorithm telling the ratio between the approximated solution and the real optimal solution. This means, the closer the approximation factor \( \rho \) is to value 1 the more exact the approximation algorithm is.

Nevertheless, worth mentioning is that there are other types of UFL problems that are classified as easy problems, for example the UFL problem \textit{with no opening costs}. This condition makes it possible to open every facility in the optimal solution and connect every city to its closest facility. The problem of finding the closest facility for every city has a complexity of \( O(n_f) \), where \( n_f \) is the number of facilities, and the complexity for finding the solution for the whole problem has a complexity of \( O(n_f n_c) \), where \( n_c \) is the number of cities. This means we can find a solution in polynomial time for this problem. However, having no opening costs is not typical for real-world problems. There are more representative with problems having opening costs for the facilities.

Next section will introduce the LP-relaxation of the BIP-formulation as a central tool in designing approximation algorithms for the metric UFL problem. It will also include some dual theory at which many approximation algorithms are based on. One of these is the 3-approximation algorithm of Vazirani that will be presented in section 4. Furthermore, the section will present some results from famous approximation algorithms achieved since the 1960’s.
3. The history of the metric UFL problem

3.1 The LP-relaxation and different methods for approximation algorithms

As pointed out in Ref. [1], the metric UFL problem formulated as a binary integer program, as in (2.1.), does not make the solution of the metric UFL problem easier. Hence, if the BIP-formulation is relaxed with a linear programming formulation (LP-relaxation), i.e. the variable restrictions are relaxed to all positive real numbers instead of being defined as binary, the problem becomes solvable in polynomial time. Therefore, the LP-relaxation of the BIP formulation is a central tool in approximation algorithms for finding an approximated optimum for the metric UFL problem.

The LP-relaxation is formulated as below,

\[
\begin{align*}
\text{minimize} & \quad \sum_{i \in F, j \in C} c_{ij} x_{ij} + \sum_{i \in F} f_i y_i \\
\text{subject to} & \quad \sum_{i \in F} x_{ij} \geq 1, \quad j \in C \\
& \quad x_{ij} \leq y_i, \quad i \in F, j \in C \\
& \quad x_{ij} \geq 0, \quad i \in F, j \in C \\
& \quad y_i \geq 0, \quad i \in F
\end{align*}
\]

Solving optimum for a LP-relaxation of the BIP-formulation gives of course a feasible solution for the LP-relaxed metric UFL problem. This solution is a feasible solution, and thus a feasible approximated optimum, for the BIP-formulation only if the solution takes binary values for all variables \(x_{ij}\) and \(y_i\). From the approximation algorithm of Vazirani we will see that the LP-relaxed optimal solution will always have binary values on all variables \(x_{ij}\) and \(y_i\) and therefore be a feasible approximated optimum for the BIP-formulation.

The optimal solution for the LP-relaxed problem will be the lower bound for the solution, i.e. the lowest value the LP-relaxation can take and still be feasible. This lower bound is solved by different methods and thus has given different design of the approximation algorithms used for finding a relaxed BIP optimum. Next, we are introducing some of these different approximation algorithms that have been important trying to find approximated solutions for the metric UFL problem, as is mentioned as central in Ref. [1, 7-9].

One often used technique in LP-relaxation approximation algorithms for finding the lower bound for the LP-relaxation is the LP-duality, where the LP-relaxed problem, called the primal problem, and the associated dual problem are considered. This method uses the special relationship between the primal and dual formulation of a problem.

The dual problem associated with the LP-relaxation of the BIP-formulation of the metric UFL problem is formulated as,
\[
\text{maximize } \sum_{j \in C} \alpha_j \quad (3.2)
\]

subject to \[\alpha_j - \beta_{ij} \leq c_{ij}, \quad i \in F, j \in C\]
\[\sum_{j \in C} \beta_{ij} \leq f_i, \quad i \in F\]
\[\alpha_i \geq 0, \quad j \in C\]
\[\beta_{ij} \geq 0, \quad i \in F, j \in C\]

In the dual program we introduce the dual variables \(\alpha_j\) and \(\beta_{ij}\) that can be interpreted as the total price paid by city \(j\), respectively the price paid by city \(j\) towards open facility \(i\). From the second set of constraints in (3.2) we can see that the sum of every payment from cities \(j \in C\) towards open facility \(i\) never exceeds the cost of open facility \(i\). This means, only those cities that want to be served by a facility that is closed, will be contributing towards its opening costs. The first set of constraints tells that the total price paid by city \(j\) has to be minor or equal to the cost of letting city \(j\) be served by facility \(i\), and the price paid by city \(j\) towards open this facility \(i\). The objective function in this dual program is to maximize the sum of the total price paid by city \(j\). What factors that make these dual variables take their respectively values, will be discussed more in section 4. In that section we will solve an example of the metric UFL problem with a method that uses this dual formulation.

With inspiration from the book of Lundgren et al [6] we now present some useful facts about the primal and the dual problem and the theory of LP-duality. The relation between the primal and the dual formulation of the problem is,

<table>
<thead>
<tr>
<th>Primal problem</th>
<th>Dual problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\min \ z = c^T x)</td>
<td>(\max \ w = b^T v)</td>
</tr>
<tr>
<td>(Ax \geq b)</td>
<td>(A^T v \leq c)</td>
</tr>
<tr>
<td>(x \geq 0)</td>
<td>(v \geq 0)</td>
</tr>
</tbody>
</table>

We know from the Weak Dual Theorem that a feasible solution in the dual problem, always gives an optimistic approximation of the optimal value of the objective function in the primal problem (LP-relaxation). A feasible dual solution will always have an objective value that is minor or equal to the real optimum value of the objective function for the LP-relaxation.

We also know that if there is a feasible solution for respectively primal and dual problem that gives the same objective value, this feasible solutions is the optimal solution for respectively problem. Thus, the optimal value for the objective functions have been found and this is the same value for the primal and dual problem.

Finally, we know that the primal and dual feasible solutions are both optimum iff these feasible solutions satisfy the complementary slackness conditions. Therefore, many approximation algorithms use approximated variants of the complementary slackness conditions. This is also explained further in section 4.
The complementary slackness conditions are the general relationship between a primal constraint $i$ and the corresponding dual variable $v_i$, saying that either the slack variable in the primal constraint has the value zero or the dual variable is zero,

$$v_i \left( b_i - \sum_{j=1}^{n} a_{ij} x_j \right) = 0 \quad \text{where } n = \text{number of primal variables} \quad (3.3)$$

And similarly, they are the general relationship between the primal variable $x_j$ and the corresponding dual constraint $j$, that say that either the slack variable in the dual constraint has the value zero or the primal variable is zero,

$$x_j \left( \sum_{i=1}^{m} a_{ij} v_i - c_j \right) = 0 \quad \text{where } m = \text{number of dual variables} \quad (3.4)$$

The primal and dual complementary slackness conditions are formulated as below, where we call the first two conditions, (S1) and (S2), for the primal conditions and (S3) and (S4) for the dual conditions,

(S1) $\forall i \in F, j \in C: \ x_{ij} > 0 \Rightarrow \alpha_j - \beta_{ij} = c_{ij}$

(S2) $\forall i \in F: \ y_i > 0 \Rightarrow \sum_{j \in F} \beta_{ij} = f_i$

(S3) $\forall j \in C: \ \alpha_j > 0 \Rightarrow \sum_{i \in F} x_{ij} = 1$

(S4) $\forall i \in F, j \in C: \ \beta_{ij} > 0 \Rightarrow y_i = x_{ij}$

In section 4.1 we will see how the 3-approximation algorithm of Vazirani uses an approximated version of the complementary slackness conditions by relaxing the primal condition (S1).

An approximation algorithm using LP-duality can be designed in two specific ways, based on Primal-dual scheme or on Dual fitting schemes. Both of these variants are based on updating the primal and dual variables along the algorithm. The update of the primal variables reflects the corresponding steps taken in the algorithm, i.e. if facility $i$ is opened the variable $y_i$ gets value 1 for this facility, and the update of the dual variables aims to provide a certain approximation guarantee. The Primal-dual scheme is explained more by the presentation of the 3-approximation algorithm of Vazirani, a well-known approximation algorithm using Primal-dual scheme.

The Dual fitting based algorithms make the updates of the primal and dual variables such that they are satisfying all complementary slackness conditions without a relaxation. Though, as the primal variables give a feasible solution to the primal problem, the dual variables are permitted to be infeasible. The degree of infeasibility of the dual variables is thus bounded. After dividing every dual variable by an appropriate factor $\gamma$ the dual is feasible in the dual LP. One often defines an additional LP-relaxation, called factor
revealing LP to be able to find the suitable factor $\gamma$ for all instances of the problem. The factor $\gamma$ corresponds to the approximation guarantee of the algorithm.

Another often used technique in LP-relaxation approximation algorithm, for finding the lower bound, is the LP-rounding. This algorithm is based on solving the LP-relaxation with a known method. Because this solution, in most of the cases is not integral, the algorithm rounds the solution to make it feasible. Though, not more than to a value that is maximum $\rho$-times the real optimal value. This makes the algorithm to an approximation guarantee of $\rho$. This method is a more expensive process than solving the LP-relaxation with LP-duality.

Another technique is the filtering technique where some variables in the BIP-formulation are fixed to zero. The challenge is to find out which variables that will be fixed so that every solution of the problem get a value of the objective function, that is within a factor of $(1 + \varepsilon)$ of the real optimum value of the original BIP-formulation. The algorithm then continues with solving a feasible solution to the filtered problem with for example rounding a fractional solution or by using any combinatorial algorithm.

3.2 Earlier results from approximation algorithm for the metric UFL

Since the beginning of the 1960’s there have been several attempts trying to approximate the metric UFL problem. People have used different techniques and approaches trying to get to the best approximation algorithms and this has lead to more or less good results. This section will present historically central approximated results for the metric UFL problem and the people behind these.

The truth is that the approximation guarantee has shown to be quite high for many of these approximation algorithms. Further, in 1997 Guha and Khuller [3] proved that the best approximation guarantee an approximation algorithm for the metric UFL problem can obtain is 1.463, as long as $NP \notin DTIME[n^{O(\log\log n)}]$. $DTIME$ is defined as the computational resource of computation time for the deterministic Turing machine. In other words it represents the number of computation steps that a “normal” physical computer would require for solving the problem. The result was further proved by Sviridenko that stated that the best ratio is 1.463 unless $P = NP$, see Ref. [2]. Therefore, the ratio 1.463 has become generally accepted of people working with approximation algorithms for the metric UFL problem.

Byrka [2] is referring to Hochbaum who presented a greedy algorithm for the general UFL-problem with an approximation guarantee of $O(\log n)$, where n is the number of cities. It is known, by a reduction from the set cover problem, that this ratio can not be improved unless $NP \in DTIME[n^{O(\log\log n)}]$.

Shmoys, Tardos and Aardal came with the first approximation algorithm with a constant approximation guarantee for the metric UFL problem. This approximation ratio was 3.16, Ref [2], and was obtained by combining LP-rounding and the filtering technique, Ref [1]. The best approximation ratio today is 1.488 obtained by Li [5] in 2011. This result beats the result of Byrka [2] at 1.5.
4. Vazirani’s Primal-dual schema based factor 3-approximation algorithm

One well-known approximation algorithm using Primal-dual scheme is the 3-approximation algorithm of Vazirani from 2001 [7]. This algorithm gives an approximation ratio at 3 and is often mentioned in other sources and articles when referring to a Primal-dual schema based approximation algorithm for the metric UFL problem. In this part we will introduce the basics for this algorithm and take a closer look at the technicalities it is based on.

As mentioned in previous sections the Primal-dual scheme algorithm is based on a linear program relaxation, formulation (3.1), of the BIP-formulation (2.1) of the metric UFL problem. The algorithm also uses the dual program, formulation (3.2), to find the optimal solution for the LP-relaxation. Recall the LP-relaxation and the dual problem formulated as,

**LP-relaxation,**

\[
\text{minimize } \sum_{i \in F, j \in C} c_{ij}x_{ij} + \sum_{i \in F} f_i y_i \\
\text{subject to } \sum_{i \in F} x_{ij} \geq 1, \quad j \in C \\
x_{ij} \leq y_i, \quad i \in F, j \in C \\
x_{ij} \geq 0, \quad i \in F, j \in C \\
y_i \geq 0, \quad i \in F
\]

**dual problem,**

\[
\text{maximize } \sum_{j \in C} \alpha_j \\
\text{subject to } \alpha_j - \beta_{ij} \leq c_{ij}, \quad i \in F, j \in C \\
\sum_{j \in C} \beta_{ij} \leq f_i, \quad i \in F \\
\alpha_j \geq 0, \quad j \in C \\
\beta_{ij} \geq 0, \quad i \in F, j \in C
\]

It is important to get an intuitive understanding of the dual problem to be able to see how it is used in the algorithm of Vazirani. In the dual program we introduced the dual variables \(\alpha_j\) and \(\beta_{ij}\) that could be interpreted as the total contribution of payment from city \(j\), respectively the contribution of payment from city \(j\) towards ope facility \(i\).
The dual variables $\alpha_j$ associates to the primal constraints formulated as,

$$\sum_{i \in F} x_{ij} \geq 1,$$

in the LP-relaxation, and the dual variables $\beta_{ij}$ are related to the primal constraints,

$$y_i - x_{ij} \geq 0.$$

The dual constraints,

$$\alpha_j - \beta_{ij} \leq c_{ij},$$

state that the difference between the total contribution from city $j$ and the contribution from city $j$ towards open facility $i$ has to be less or equal to the cost of connecting city $j$ to facility $i$. From the deeper analysis of the algorithm in section 4.2, it will be clear that these conditions make the dual variables $\beta_{ij}$ nonzero. The dual constraints,

$$\sum_{j \in C} \beta_{ij} \leq f_i,$$

state that the total contribution of every city $j$ towards opening facility $i$ has to be less or equal to the costs of opening facility $i$. This condition will also be central in the algorithm in Phase 1, one of the two phases that the algorithm is based on.

Before we go on to the next section we summarize the variables defined in the algorithm, in table 4.1 below:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C$</td>
<td>Set of demand points (cities)</td>
</tr>
<tr>
<td>$F$</td>
<td>Set of possible facility locations</td>
</tr>
<tr>
<td>$\phi(j) = i$</td>
<td>The assignment function, city $j$ is connected to facility $i$</td>
</tr>
<tr>
<td>$c_{ij}$</td>
<td>Service or connection cost for assigning city $j$ to facility $i$</td>
</tr>
<tr>
<td>$f_i$</td>
<td>Opening cost for facility $i$</td>
</tr>
<tr>
<td>$x_{ij}$</td>
<td>Indicator variable telling if city $j$ is connected to facility $i$</td>
</tr>
<tr>
<td>$y_i$</td>
<td>Indicator variable telling if facility $i$ is opened</td>
</tr>
<tr>
<td>$\beta_{ij}$</td>
<td>The price paid by city $j$ towards opening facility $i$</td>
</tr>
<tr>
<td>$\alpha_j$</td>
<td>The total price paid by city $j$</td>
</tr>
</tbody>
</table>

Table 4.1. The variables used in the approximation algorithm of Vazirani
4.1 The relaxed complementary slackness conditions

In other sections we have already mentioned the central role that the complementary slackness conditions have in Primal-dual schemes, for finding the optimum of the LP-relaxation, i.e. the approximated optimal value for the metric UFL problem. As many other approximation algorithms the algorithm of Vazirani [7] is built on approximated complementary slackness conditions. In this case, the algorithm of Vazirani adapts to a relaxed version of the primal complementary slackness conditions. This differs the algorithm from other primal-dual algorithms where it is more common to relax the dual complementary slackness conditions. We know from previous section that the primal and dual feasible solutions are both optimum iff they satisfy all the complementary slackness conditions. Significant for the approximation algorithm of Vazirani [7,9] is that a primal and dual optimum is found, if the primal and dual feasible solutions satisfy all complementary slackness conditions as presented in section 3.1, besides from condition (S1). Instead, the algorithm will satisfy a relaxed version of conditions (S1) that is, 

\[ x_{ij} > 0 \implies (1/3)c_{ij} \leq \alpha_j - \beta_{ij} \leq c_{ij} \]

The approximation guarantee of 3 will follow from the approximation of the complementary slackness conditions above. Because all other complementary slackness conditions are satisfied, every primal and dual feasible solution are related as the value of the dual objective function is less or equal to the value of the primal objective function that, in turn, is less or equal to 3 times the value of the dual objective function. Thus, the dual objective function value \( \leq \) the primal objective function value \( \leq 3 \) (the dual objective function value), which implies that the approximation guarantee is 3.

4.2 The algorithm

In this section we will present the algorithm in detail. Overall, we can say that the algorithm is based on updating the primal and dual variables along the working-time. As stated in Ref. [9] the updates of the variables aim to satisfy more and more of the complementary slackness conditions (S2), (S3), (S4) and the relaxed version of (S1). The update of the primal variables reflects the corresponding steps taking in the algorithm, i.e. if facility \( i \) is opened the variable \( y_i \) gets value 1 for this facility, and the update of the dual variables aims to provide a certain approximation guarantee, in this case a 3-approximation guarantee.

Significant for primal-dual schemas is that these algorithms yields good running times. According to Vazirani [7] the algorithm has the running time \( O(m \log m) \) where \( m \) is the total number of edges. This is due to the fact that the algorithm will consider every edge at most two times – the first time when it becomes tight and the second time when the city gets connected. This gives \( O(\log n_f) \) for each of these two considerations, where \( n_f \) is defined as the number of facilities.

The algorithm starts by relaxing the primal complementary slackness condition (S1) as described in section 4.1. Then it goes on with the main part of the algorithm, i.e. the two phases. This is where the update of the primal and dual variables will take place and can be perceived as the most technical part of the algorithm. Phase 1 aims to find a feasible solution to the dual problem, a set of temporary open facilities \( F_t \), and a set of tight edges. Phase 2 will choose a subset of the temporary open facilities to be permanently open and
will also find the optimal assignment – the optimal connection between cities and open facilities. The two phases will now be described more in detail based on Ref. [4, 7, 9]. In next section the algorithm will be applied on an example.

Phase 1
The ambition is to find as large a dual solution as possible and therefore this phase will be built on successively raising the dual variables. In the beginning the dual variables $\alpha_j$ and $\beta_{ij}$ are all zero, every facility $i$ is closed and every city $j$ is unconnected. To be able to associate the events in Phase 1 with something a notion of time, that start at zero, is defined.

The phase begins with raising the dual variables $\alpha_j$ uniformly in unit time, for every city $j$ that is not yet connected to any facility. The raising of the $\alpha_j$ for city $j$ stops when it reaches the same value as the connection cost for this city $j$ to a facility $i$, i.e. when $\alpha_j = c_{ij}$ for city $j$. This can be interpreted as city $j$ has contributed enough to reach facility $i$. The edge $(i,j)$ for which $\alpha_j = c_{ij}$ will be called a tight edge.

At this point the algorithm will look whether facility $i$ from this tight edge $(i,j)$ is temporarily open or not. A facility will be declared temporarily open if $\sum_{j \in C} \beta_{ij} = f_i$, which means that the total sum of payments from cities $j \in C$ towards open facility $i$ has to be equal to the cost of opening facility $i$. From the beginning of Phase 1 every facility is closed, but during Phase 1 there will be more and more facilities that are declared as temporarily open.

If facility $i$ is not temporarily open, the dual variable $\beta_{ij}$ will henceforth be raised uniformly with the same unit rate as $\alpha_j$ to ensure that the constraint $\alpha_j - \beta_{ij} \leq c_{ij}$ in the dual problem (3.2) is not violated. Edges $(i,j)$ such that $\beta_{ij} > 0$ will be called special. This event will help facility $i$ towards being temporarily opened, because $\beta_{ij}$ will contribute for facility $i$ to be opened by going from zero to a higher positive value.

If facility $i$ is temporarily open the tight edge $(i,j)$ will be declared as used, i.e. city $j$ will be connected to facility $i$ and the rising of the dual variables $\alpha_j$ and $\beta_{ij}$ will be stopped. In fact, every unconnected city $j$ that have reached facility $i$, i.e. have a tight edge to facility $i$, are connected to that facility at the time when it gets temporarily open. After this, facility $i$ will be declared as the connecting witness for each of these cities. The event of connecting cities with tight edges to temporarily open facilities, will continue like this during Phase 1. When all cities have been connected the algorithm will continue with Phase 2.

Note that the events in Phase 1 are designed to obtain the complementary slackness conditions (S1) and (S2). That is why the dual variables $\beta_{ij}$ are raised for some cities and facilities. However, in most cases when $\beta_{ij}$ is raised it is at the expense of condition (S4) being violated because the same city $i$ have raised its dual variable $\beta_{ij}$ to more than one facility, i.e. to more facilities than the one it is connected to. Therefore, Phase 2 needs to be performed so that (S4) can be satisfied for all cities and facilities. It will then end up satisfying all complementary slackness conditions (S2), (S3), and (S4) in its initial form and the relaxed version of (S1).
Phase 2

Due to the fact that the same city may have paid towards temporarily open several facilities in Phase 1, too many facilities may have been temporarily opened. To avoid this, Phase 2 will select a subset of temporarily open facilities to be opened permanently.

Phase 2 starts with creating a subgraph $T$ of the original graph $G$ that consists of all the special edges that come up in Phase 1. A graph $T^2$ is also defined consisting of edges $(u,v)$ if there is a path between $u$ and $v$ in subgraph $T$ of length at most 2. A last subgraph $H$ is defined as the subgraph of $T^2$ induced on $F$ and this will be central in this phase.

Besides from the fact that subgraph $H$ consists of the special edges from $T^2$ that goes to temporarily open facilities, it also consists of edges between the facilities that are declared as conflicting. Temporarily open facilities are conflicting if the same city $j$ has contributed payment to them in order to be temporarily opened, that is both $\beta_{ij}$ and $\beta_{i'j}$ got a value $>0$ in Phase 1. If there are two conflicting facilities $i$ and $i'$ there will be an edge between these two in subgraph $H$.

There is also a central subset $I$ which is defined as the maximal independent subset of the facilities from subgraph $H$. The order in which the facilities in subgraph $H$ will be selected into the subset depends on which order they were temporarily opened in Phase 1. All facilities in $I$ will be declared as permanently open and is the set of facilities that the cities in Phase 2 can be connected to. Since $I$ is a maximal independent subset it will be impossible to put both $i$ and $i'$ in $I$, due to the edge between $i$ and $i'$ in $H$. This ensures that city $j$ only contributes to opening the facility that it will be connected to, and this means only one of $\beta_{ij}$ and $\beta_{i'j}$ will be $>0$.

The main event in Phase 2 is then to directly or indirectly connect every city $j$ to an open facility $i$ that is in $I$. A city $j$ is directly connected to facility $i$ if $(i,j)$ is a tight edge $\alpha_j = c_{ij}$ and facility $i \in I$. Then we set $\phi(j) = i$, which says that city $j$ is declared to be connected to facility $i$, and $\beta_{i'j} = 0$ which means city $j$ will not pay towards opening any other facility than $i$.

If $(i,j)$ is a tight edge but $i \notin I$ we know that the closing witness to $i$ is $i'$, in other words $i'$ is the neighbour of $i$ in $H$, and therefore we can say that $i'$ is in subset $I$. This makes city $j$ indirectly connected to $i'$, $\phi(j) = i'$, and $i$ will be seen as the connecting witness of city $j$ in Phase 1.

When every city $j$ has been connected in Phase 2 an optimal solution for the LP-relaxation and the approximated optimal value for the metric UFL problem is obtained, defined by the subset $I$ and the $\phi$ – function. More precisely $x_{ij} = 1$ iff $\phi(j) = i$ and $y_i = 1$, otherwise they will be zero.

4.3 Analysis and example

It can be hard to see all technicalities in the algorithm of Vazirani without following an example. Although, explicit examples of the algorithm are rare in the literature. Perhaps this is because small problems appear too simple and will be better solved with a greedy method. Thus, for easy solvable problems there is unnecessary to use this approximation algorithm by Vazirani. At the same time, when the metric UFL problem is too complicated, with a large number of facilities and cities, the problem perceives to be too complex to solve by hand.
However, we need a simple example to demonstrate the technicalities in the algorithm. The example has to be quite small with a quite obvious optimal solution. Therefore, we will use the following example with two possible facilities and three cities – thus a very simple problem.

Consider the following situation; a bipartite graph with bipartition \((F,C)\) where \(F\) is the set of two potential facilities \(i\) and \(C\) is the set of three cities \(j\) that have to be served.

![Bipartite Graph](image)

Every facility is uncapacitated and has an opening cost \(f_i\). Every customer has a service cost \(c_{ij}\) identified for every facility. Also, the facilities have an opening cost. The problem is metric due to the metric condition \(c_{ij'} \leq c_{ij'} + c_{ij}\) that is satisfied for every pair of nodes \((i,j')\) in the graph. Thus, we want to find out which of these facilities that has to be opened and which city to be served from which facility for an optimal solution with the approximation algorithm. It is easy to see that the optimal solution is to open facility 1 and assign every customer to that facility, because it gives a lower cost than opening facility 2, or both facility 1 and 2, and assign the cities according to these. Opening facility 1 gives a total cost of 16, opening facility 2 gives an total cost of 18 and opening both facilities gives a cost of 17. The BIP-formulation, LP-relaxation and the dual problem look like before as in equation (2.1), (3.1) and (3.2). When the algorithm has relaxed the complementary slackness condition (S1) it goes on with Phase 1.

**Phase 1**

Recall that the dual variables \(\alpha_j\) and \(\beta_{ij}\) are all zero, every facility \(i\) is closed and every city \(j\) is unconnected and a notation of time, \(t\), that start at zero, is defined.

1) The algorithm starts to raise the dual variables \(\alpha_j\) for every unconnected city \(j\) until \(\alpha_1 = \alpha_2 = \alpha_3 = 2\).

At this rate the algorithm will stop raising the dual variables because \(\alpha_3\) and \(\alpha_3\) have reached the same value as the connection cost for city 1 to a facility \(i\) and for city 3 to a facility \(i\),

\[
\alpha_1 = 2 = c_{11} \text{ and } \alpha_3 = 2 = c_{13},
\]

which makes (1,1) and (1,3) tight edges.
At this point the algorithm first looks whether facility 1 is opened, and because \( \sum_{i \in C} \beta_{1i} \leq f_1 \) it is not opened. Thus, as \( \alpha_1 \) continue to raise next time, the dual variable \( \beta_{11} \) will be raised from 0 with the same unit rate as \( \alpha_1 \), so that the dual constraint \( \alpha_1 - \beta_{11} \leq c_{11} \) wont be violated.

Then again, the algorithm looks whether facility 1 is opened, and because it is still not opened, as \( \alpha_1 \) continue to raise next time, the dual variable \( \beta_{13} \) will be raised from 0 with the same unit rate as \( \alpha_1 \), so that the dual constraint \( \alpha_1 - \beta_{13} \leq c_{13} \) will not be violated.

\[ \text{ii)} \] The algorithm continues to raise the dual variables \( \alpha_j \) for every unconnected city \( j \) until \( \alpha_1 = \alpha_2 = 3 \). At the same time each of \( \beta_{11} \) and \( \beta_{13} \) raise to value 1, i.e. \( (1,1) \) and \( (1,3) \) become special edges. This gives a contribution of 2 towards the opening cost of facility 1, \( f_1 \), i.e. \( \beta_{11} + \beta_{13} = 2 < f_1 \).

The dual variables \( \alpha_j \) stop being raised at \( \alpha_1 = \alpha_2 = 3 \) because \( \alpha_1 \) has reached the same value as the connection cost for city 1 to a facility \( j \),

\[ \alpha_1 = 3 = c_{21} \text{, which makes (2,1) a tight edge.} \]

The algorithm looks whether facility 2 is opened, and because \( \sum_{j \in C} \beta_{2j} \leq f_2 \) it is closed. Thus, as \( \alpha_1 \) continues to be raised, the dual variable \( \beta_{21} \) will be raised from 0 with the same unit rate as \( \alpha_1 \), so that the dual constraint \( \alpha_1 - \beta_{21} \leq c_{21} \) will not be violated.

\[ \text{iii)} \] The algorithm continues to raise the dual variables \( \alpha_j \) for every unconnected city \( j \) and when \( \alpha_1 = \alpha_2 = \alpha_3 = 4 \) the dual variables \( \beta_{21} \) and \( \beta_{11} \) have been raised with the same unite rate to \( \beta_{11} = 2 \) and \( \beta_{21} = 1 \) (\( \beta_{13} \) is still at value 1). Note \( (2,1) \) becomes a special edge. This gives a contribution of 1 towards the opening cost of facility 1, \( f_1 \), i.e. \( \beta_{11} + \beta_{13} = 3 = f_1 \) which makes facility 1 to be temporarily opened, \( y_1 = 1 \).

Due to that facility 1 is declared temporarily opened, all unconnected cities \( j \) with tight edges to facility 1 are connected to this facility. The tight edges to facility 1 are \( (1,1) \) and \( (1,3) \), thus \( x_{11} = x_{13} = 1 \) and the dual variables \( \alpha_1, \alpha_3, \beta_{11} \) and \( \beta_{13} \) stop at respective value and will not be further raised.

Because no connection cost has value 4 the algorithm continues to raise the dual variable \( \alpha_2 \) to value 5 where it stops due to \( \alpha_2 = 5 = c_{23} = c_{22} \), i.e. \( (2,3) \) and \( (2,2) \) become tight edges.

As \( \sum_{j \in C} \beta_{2j} = 1 \leq f_2 \), facility 2 is still closed and the dual variable \( \beta_{22} \) and \( \beta_{23} \) will be raised as \( \alpha_2 \) continues to raise.

The algorithm continues to raise \( \alpha_2 \). When \( \alpha_2 = 6, \beta_{23} = 1 \) and \( \beta_{22} = 1 \), i.e. \( (2,3) \) and \( (2,2) \) have become special. This adds a contribution of 2 towards the opening cost of facility 2, thus \( \beta_{21} + \beta_{23} + \beta_{22} = 3 < f_2 \).
When $\alpha_2 = 7$, $\beta_{23} = 2$ and $\beta_{22} = 2$ which makes facility 2 temporarily opened, $\beta_{21} + \beta_{23} + \beta_{22} = 5 = f_2$, $y_2 = 1$. This makes all unconnected cities with tight edges to facility 2 being connected to facility 2. The only unconnected city with tight edge to facility 2 is city 2, thus $x_{22} = 1$.

Phase 1 is now finished because all cities are connected to temporarily opened facilities. $F_t = \{1, 2\}$. Tight edges are $(1, 1)$, $(1, 3)$, $(2, 1)$, $(2, 3)$ and $(2, 2)$, and all these are also special edges. Accordingly, city 1 and city 3 are both contributing towards the opening of facility 2 even though they are connected to facility 1. This will be fixed in Phase 2.

Phase 2
The phase will select a subset of the temporarily opened facilities $F_t = \{1, 2\}$ to be permanently opened and then directly or indirectly connect every city $j$ to the permanently opened facility.

i) First a subgraph $T$, ($T^2$), is created that contains all special edges from Phase 1,

![Image 1](image1.png)

A subgraph $H$ is then created that contains all special edges from subgraph $T$ that goes to temporarily open facilities, $F_t = \{1, 2\}$, and an edge between the conflicting facilities. Thus, facility 1 and 2 are conflicting, subgraph $H$ looks like,

![Image 2](image2.png)
Then, the maximal independent subset $I$ is defined as $I = \{1\}$. This because facility 1 was declared temporarily opened before facility 2 in Phase 1. Thus, facility 1 is now declared as *permanently opened* and facility 2 is closed, $y_1 = 1$ and $y_2 = 0$.

The algorithm continues with directly or indirectly connecting the cities $j = \{1,2,3\}$ to the permanently opened facility 1.

Because $(1,1)$ and $(1,3)$ are tight edges city 1 and city 3 are directly connected to facility 1. Thus, $\phi(1) = 1$ and $\phi(3) = 1$, which means $x_{11} = 1$, $x_{21} = 0$, $x_{13} = 1$ and $x_{23} = 0$, and $\beta_{21}$ and $\beta_{23}$ get value zero.

City 2 gets indirectly connected to facility 1 because it has no tight edge to a facility in subset $I$, i.e. a permanently opened facility. $\phi(2) = 1$ and $x_{12} = 1$, $x_{22} = 0$.

Phase 2 is now finished because every city $j$ is connected. The primal and dual objective function values are respectively,

$$c_{11}x_{11} + c_{13}x_{13} + (c_{22} + c_{21} + c_{11}) + f_1y_1 = 2 + 2 + (5 + 3 + 2) + 3 = 17$$

and

$$\alpha_1 + \alpha_2 + \alpha_3 = 4 + 10 + 3 = 17.$$ 

Hence, the primal and dual problem gets the same optimal value 17 with the approximation algorithm by Vazirani. The approximated optimal solution for the metric UFL example is then 17, i.e. $\frac{17}{16}$ times worse than the real optimal value at 16. In this case the ratio between the approximation and the optimal solution is $\rho = 1.0625$, which is much less than the guaranteed factor 3.
5. References


e-mail: amandafors@email.com