

# Facility Location Problems: A Parameterized View <sup>\*</sup>

Michael Fellows<sup>1</sup> and Henning Fernau<sup>1,2</sup>

<sup>1</sup> The University of Newcastle  
University Drive, Callaghan, NSW 2308, Australia  
mfellows@newcastle.edu.au

<sup>2</sup> Universität Trier, FB IV—Abteilung Informatik, 54286 Trier, Germany,  
fernau@uni-trier.de

**Abstract.** Facility Location can be seen as a whole family of problems which have many obvious applications in economics. They have been widely explored in the Operations Research community, from the viewpoints of approximation, heuristics, linear programming, etc. We add a new facet by initiating the study of some of these problems from a parametric point of view. Moreover, we exhibit some less obvious applications of these algorithms in the processing of semistructured documents and in computational biology.

## 1 Introduction

The basic task of all variants of facility location problems is the following: a company wants to open up a number of facilities to serve their customers. Both the opening of a facility at a specific location and the service of a particular customer through a facility incurs some cost. The goal is to minimize the overall cost associated to a specific way of opening up facilities and serving customers. We will formalize this task in the following more precisely.

*Definitions.* We study the following problem of FACILITY LOCATION and variants thereof:

**Given:** A bipartite graph  $B = (F \uplus C, E)$ , consisting of a set  $F$  of potential facility locations, a set  $C$  of customers, and an edge relation  $E$ , where  $\{f, c\} \in E$  indicates that  $c$  can be served from the facility (at)  $f$ ; and weight functions  $\omega_F : F \rightarrow \mathbb{N}_{\geq 1}$  and  $\omega_E : E \rightarrow \mathbb{N}_{\geq 1}$  (both called  $\omega$  if no confusion may arise),  $k \in \mathbb{N}$

**Question:** Is there a set  $F' \subseteq F$  of facility locations and a set  $E' \subseteq E$  of ways to serve customers such that ((1)  $\forall f \in F (f \in F' \iff \exists e \in E' (f \in e))$ ), (2)  $\forall c \in C \exists e \in E' (c \in e)$ , and (3)  $\sum_{f \in F'} \omega_F(f) + \sum_{e \in E'} \omega_E(e) \leq k$ ?

The first condition links the choice of facility locations with the choice of customer services, while the second condition expresses the necessity that every customer can be served through the choice of facility locations. The third

---

<sup>\*</sup> This research has been supported by the Australian Research Council through the Australian Centre of Excellence in Bioinformatics.

condition formalizes that the overall costs (weights) of opening up facilities and serving customers should be bounded by  $k$ . In the literature, the problem formulated above is mostly known as UNCAPACITATED DISCRETE FACILITY LOCATION PROBLEM, see [3] for a good recent overview. Notice that we subsumed the usually separated customer demands and service costs (per demand unit) into the weight function  $\omega_E$ . We will also discuss variants (e.g., allowing real number costs) in this paper.

Alternatively, and sometimes more convenient, this problem can be formulated in terms of a “matrix problem:”

FACILITY LOCATION (MATRIX FORMULATION)

**Given:** A matrix  $M \in \mathbb{N}_{\geq 1}^{(n+1) \times m}$ , indexed as  $M[0 \dots n][1 \dots m]$ ,  $k \in \mathbb{N}$

**Question:** Is there a set  $C \subseteq \{1, \dots, m\}$  of columns and a function  $s : \{1, \dots, n\} \rightarrow C$  such that  $\sum_{f \in C} (M[0, f] + \sum_{c: s(c)=f} M[c, f]) \leq k$ ?

In the matrix formulation, the columns play the role of the potential facility locations and the rows represent the customers to be served (except for row 0). Since “missing edges” in the bipartite graph formulation can be expressed as edges which have “infinite weight” (corresponding in turn to a weight larger than  $k$  in the decision problem formulation), we can assume that the graph is indeed a complete graph. Then, the matrix  $M[1 \dots n][1 \dots m]$  contains the weights of the edges, while  $M[0][1 \dots m]$  stores the weights associated to potential facility locations.

In the following, we will use terminology from both formulations interchangeably, according to convenience.

*Fixed parameter tractability.*  $\mathcal{NP}$ -hard computational problems are ubiquitous in economics. One approach to overcoming this difficulty is to devise algorithms that can solve arbitrary instances of such a problem under the restriction that a certain entity, called the parameter, is small. This concept is usually formalized as follows: Problem instances are elements of  $\Sigma^* \times \mathbb{N}$ , and an instance  $I = (w, k)$  is to be decided in time  $\mathcal{O}(p(|w|)f(k))$ , where  $p$  is a polynomial (whose degree does not depend on the parameter  $k$ ) and  $f$  is an arbitrary function. Problems that can be solved within such a time restriction are called *fixed parameter tractable*, or in  $\mathcal{FPT}$ , for short. Equivalently, a problem is in  $\mathcal{FPT}$  iff there exists a poly-time computable self-reduction that maps an instance  $I = (w, k)$  onto an (other) instance  $I' = (w', k')$  of the same problem whose overall size is limited by a function  $g(k)$ , i.e.,  $|w'| + k' \leq g(k)$ . Then,  $I'$  is also called a *problem kernel* for  $I$ . There is also a complementing hardness theory in this field, basically reflected by the so-called W-hierarchy  $\mathcal{FPT} \subseteq W[1] \subseteq W[2] \subseteq \dots$ , where  $W[1]$ -hardness is the notion that corresponds best to  $\mathcal{NP}$ -hardness in classical complexity theory. Further details can be found in the textbook [8].

*Our contribution.* Facility location problems have been quite extensively studied in the literature under many perspectives (hardness, approximation, heuristics, etc.), and this is of course also thoroughly done within the operations research and management science communities. Our aim is to initiate systematic research on these problems from the viewpoint of parameterized complexity. We will show

parameterized tractability for the described problem formulation and many variants thereof, but also give some related intractability results. In doing so, we also provide an introduction of how  $\mathcal{FPT}$ -results can be obtained in a rather systematic fashion. Moreover, we will exhibit connections to many application areas outside economics, which might stir up research between different communities on this topic.

*Related work and variations.* It is often useful to relate the costs of serving customers with an underlying metric space, i.e.,  $\omega_E(c, f)$  can be described by the distance of  $c$  and  $f$  times the demand that is incurred by  $c$ . This problem variant is also referred as the (metric) uncapacitated facility location problem. A more general setting is given in the capacitated facility location problem, where each facility can only serve up to a given maximum load; then, there are again two variants considering the possibility that a demand of one specific customer might be satisfied from only one facility or possibly split among different facilities (which is of course more flexible). Good overviews of approximation algorithms for many variants can be found in [3, 13, 14].

## 2 FACILITY LOCATION is in $\mathcal{FPT}$

We describe several approaches to the statement of the headline. This can be also taken as a short introduction to the whole field of parameterized algorithmics in a nutshell, along with this specific example.

One of the first things that has to be decided is what parameters (as a secondary measurement of the whole input) are to be considered. In our case, natural choices could be: (1) the number  $n$  of customers, (2) the number  $m$  of potential facility locations, (3) an upperbound  $k$  on the cost, or (4) an upperbound  $\ell$  on the number of facilities that could be opened.

A trivial brute-force approach immediately yields:

**Theorem 1.** FACILITY LOCATION can be solved in time  $\mathcal{O}^*(2^m)$ .

Parameter  $m$  is not particularly interesting, as long as we are mainly interested in classification results. For the other parameters suggested, the situation is less clear and will be discussed in the remainder of this paper. We put special emphasis on  $k$ , since this is the choice of parameter that is usually considered natural for minimization problems. The  $\mathcal{O}^*$ -notation suppresses polynomial factors.

### 2.1 Finding reduction rules

Reduction rules are heuristics that help shrink the search space. Often, a collection of reduction rules define a so-called kernelization, i.e., a poly-time self-reduction of a problem instance  $(I, k)$  to  $(I', k')$  such that the size of the new instance  $(I', k')$  is only upperbounded by a function in  $k$ . This type of reductions is crucial for the  $\mathcal{FPT}$  methodology since it characterizes the class  $\mathcal{FPT}$ . So, the quest for finding reduction rules is essential for parameterized algorithmics.

The following observation is easy but crucial:

**Reduction Rule 1** *If a given instance  $(M, k)$  with  $M \in \mathbb{N}_{\geq 1}^{(n+1) \times m}$  obeys  $n > k$ , then return NO.*

**Lemma 1.** *Rule 1 is valid.*

*Proof.* Each customer must be served. Since edges have positive integer weights, each customer thus incurs “serving costs” of at least one unit. Hence, no more than  $k$  customers can be served.  $\square$

**Lemma 2.** *After having exhaustively applied Rule 1, the reduced instance  $(M, k)$  will have no more than  $k$  rows.*

Notice that the preceding lemma builds a close link between the parameters  $k$  and  $n$ .

A facility location  $f$  is described by the vector  $v_f = M[0 \dots n][f]$ . These vectors can be compared componentwisely.

**Reduction Rule 2** *If for two facility locations  $f$  and  $g$ ,  $v_f \leq v_g$ , then delete  $g$ ; the parameter stays the same.*

**Lemma 3.** *Rule 2 is sound.*

*Proof.* Obviously, a solution to the Rule-2-reduced instance is also a solution to the original instance. If we had a solution  $S$  to the originally given instance which contains a facility  $g$  and there is another facility  $f$  such that  $v_f \leq v_g$ , then a solution  $S'$  obtained from  $S$  by choosing facility  $f$  instead of  $g$  (and choosing to serve any customer served by  $f$  in  $S$  to be served by  $g$  in  $S'$ ) will also be a valid solution which comes at no greater cost. This way, we can gradually transform  $S$  into a solution which is also a valid solution to the Rule-2-reduced instance and has no larger costs than  $S$ .  $\square$

## 2.2 Kernelization through well-quasi orderings

Central to the complexity class  $\mathcal{FPT}$  is the concept of kernelization, since it characterizes  $\mathcal{FPT}$ . We first provide a “quick classification” of FACILITY LOCATION in  $\mathcal{FPT}$ , based on well-quasi orderings.

**Theorem 2.** FACILITY LOCATION *is fixed-parameter tractable.*

*Proof.* We show that, after having exhaustively applied Rules 1 and 2, we are left with a problem of size  $f(k)$ . Let  $(M, k)$  be reduced with respect to Rules 1 and 2. By Lemma 2, we know that  $M$  contains no more than  $k$  rows, since  $(M, k)$  is Rule-1-reduced. Each facility is therefore characterized by a  $(k + 1)$ -dimensional vector. Since  $(M, k)$  is Rule-2-reduced, all these vectors are pairwise uncomparable. According to Dickson’s Lemma they could be only finitely many, upperbounded by some function  $g(k)$ . Hence,  $M$  is a matrix with no more than  $(k + 1)g(k)$  entries.  $\square$

The function  $f(k)$  derived for the kernel size in the previous theorem is huge, yet it provides the required classification.

### 2.3 Kernelization refinements

To obtain a better algorithm, observe that a solution can be viewed as a partition of the set of all customers into groups such that customers within the same group get served by the same facility. In actual fact, a solution specified by the selected facilities and selected serving connections can be readily transformed into this sort of partition. Also the converse is true: given a partition of the set of customers, we can compute in polynomial time which the cheapest way to serve this group by a certain facility is, so that an optimal solution (given the mentioned partition) in the sense of specifying the selected facilities and the chosen serving connections can be obtained.

This model immediately allows to derive the following result:

**Lemma 4.** *On an instance  $(M, k)$ , where  $M \in \mathbb{N}_{\geq 1}^{(n+1) \times m}$ , FACILITY LOCATION can be solved in time  $\mathcal{O}(k^k p(g(k)) + nm)$ , where  $p$  is some polynomial and  $g(k)$  bounds the number of facilities.*

*Proof.* The kernelization rules 1 and 2 can be applied in time  $\mathcal{O}(nm)$ . Then, we have to check all partitions; there are actually  $o(k^k)$  many of them (more precisely, this is described by Bell's number whose asymptotics is due to de Bruijn (1958), see <http://mathworld.wolfram.com/BellNumber.html>). For each partition, we have to compute its cost incurred by the assumption that each group in the partition is served by one facility. Hence, per partition  $p(g(k))$  computations have to be performed.  $\square$

Still, this algorithm is practically useless due to the huge constants. Let us now develop a better algorithm. Let us first focus on kernelization.

**Reduction Rule 3** *Consider an instance  $((B, \omega_E, \omega_F), k)$  of FACILITY LOCATION. The following modifications will not affect the parameter.*

**Forall facilities  $f$  do**

(1) *If  $\omega_F(f) \geq k$ , then delete  $f$ .*

**Forall customers  $c$  do**

(2) *If  $\omega_F(f) + \omega_E(c, f) > k + 1$ , then set  $\omega_E(c, f) := k + 1 - \omega_F(f)$ .*

The following is obvious.

**Lemma 5.** *Rule 3 is sound.*

**Lemma 6.** *An instance  $(M, k)$ ,  $M \in \mathbb{N}_{\geq 1}^{(n+1) \times m}$ , of FACILITY LOCATION obeys  $nm \leq (k + 1)^{k+2}$  if it is reduced with respect to Rules 1, 2 and 3.*

*Proof.* Let  $(M, k)$  be such a reduced instance. Due to Rule 1, there are at most  $k$  customers. Due to Rule 3, there are at most  $(k + 1)^{k+1}$  different facility vectors. Due to Rule 2, vectors of different facilities must be different. Hence, there are at most  $(k + 1)^{k+1}$  facilities. Since each facility vector has at most  $k + 1$  entries,  $M$  has no more than  $(k + 1)^{k+2}$  many entries.  $\square$

The estimate of the preceding lemma is probably not very sharp, since Rule 2 was used in a very rough way; the number of facilities is rather upperbounded by the maximum number of antichains in the space of  $(k + 1)$ -dimensional vectors with entries from 1 to  $k + 1$ .

## 2.4 Improving on brute force by dynamic programming

The idea of “dynamic programming on subsets” improves the running time.

---

**Algorithm 1** A dynamic programming algorithm for facility location: FLdp

---

**Require:** a bipartite graph  $B = (F \uplus C, E)$  with weights  $\omega_E$  and  $\omega_F$

**Ensure:** an implicit facility location strategy incurring minimum weight

```

if  $|C| > k$  then
  return NO;
 $s(\emptyset) := 0$ ;
for all  $X \subseteq C$  do
  compute  $os(X)$ ; {in time  $\mathcal{O}(|X| \cdot |F|)$ }
for  $i := 1, \dots, |C|$  do
  for all  $X \subseteq C, |X| = i$  do
     $s(X) := \min_{\emptyset \subsetneq Y \subsetneq X} (os(Y) + s(X \setminus Y))$ 
    {Every customer belongs either to  $Y, X \setminus Y$  or to  $C \setminus X$ ;}
    {Convention:  $\min_{\emptyset} \dots = \infty$ .}

```

---

**Theorem 3.** FACILITY LOCATION can be solved in time  $\mathcal{O}(2^k m + 3^k)$  on a given instance  $(M, k)$  with  $M \in \mathbb{N}_{\geq 1}^{(n+1) \times m}$ .

*Proof.* We start with Rule 1. In a preprocessing phase, we compute the cost incurred by a certain set of customers when being served by a single facility, storing the results in a table “one-serve” ( $os$ ) with  $2^k$  entries. This also includes the costs for opening up that facility. Then, we can compute the minimal costs of serving a certain group of customers by some facilities by dynamic programming, combining two subsets at a time. If we have stored the results of the preprocessing in table  $os$ , each step in the dynamic programming will take only constant time, so that we arrive at the following formula for dynamic programming:

$$s(X) := \min_{\emptyset \subsetneq Y \subsetneq X} (os(Y) + s(X \setminus Y)) \quad (1)$$

This amounts in  $\mathcal{O}(3^k)$  operations. Namely, there are basically three possibilities for a customer: it either belongs to  $Y, X \setminus Y$  or to  $C \setminus X$ .  $\square$

*Remark 1.* Notice that the bound  $k > n$  immediately implies an  $\mathcal{O}^*(3^n)$  estimate for Alg. 1. Alternatively, we can first kernelize (with respect to  $k$ ) and then run Alg. 1. on the kernel, which would amount in an  $\mathcal{O}^*((2k)^k)$  estimate due to Lemma 6.

## 2.5 Further improvements

In a very recent paper, A. Björklund, T. Husfeldt, P. Kaski and M. Koivisto [1] described how to further speed up dynamic programming on subsets in just

a situation as encountered with our problem. Their approach is based on fast subset convolution, also known as (fast) Möbius transform. More specifically, the recursion from Eq. (1) can be seen as a subset convolution (in the min-sum semiring). Let us shortly explain this approach (with  $n$  customers and  $m$  facilities). By computing first the ranked Möbius transform (which basically takes  $2^n$  steps), this convolution operation can be performed in the transformed space with  $2n + 1$  operations, and the inverse Möbius transform is of similar cost as the Möbius transform itself.

**Corollary 1.** *If all the integer weights lie between 1 and  $N$  for a given instance of MINIMUM FACILITY LOCATION, the problem can be solved in time  $\mathcal{O}(2^n n^3 (nm)^2 \log(N))$ .*

The assumption of a bounded range of integer weights is not so unrealistic in many scenarios; for example, in our parameterized setting, we can assume (by our reduction rules) that those weights lie between 1 and  $k + 1$ .

### 3 Variants of FACILITY LOCATION

*Median / Means Problem.* The  $k$ -MEDIAN PROBLEM and the  $k$ -MEANS PROBLEM are defined just as the FACILITY PROBLEM, except for the fact that there are no costs for opening up facilities (and mostly, it is required that exactly  $k$  facilities should open). Means and median problems only differ in the metric, i.e., the way point distances are measured (sums of squares of distances vs. sums of distances). All corresponding problems are  $\mathcal{NP}$ -hard, even for Euclidean spaces. Obviously, these problems have quite a geometric flavor. Again, it is not hard to see by a simple analysis of our results of the preceding section that also the  $k$ -MEDIAN PROBLEM and the  $k$ -MEANS PROBLEM are in  $\mathcal{FPT}$ : namely, even the common generalization of these problems, viewed as variants of FACILITY LOCATION, where the opening cost of a facility might be zero, is in  $\mathcal{FPT}$ .

Many approximation algorithms are known for this problem, see [5]. Notice that often (also there) it is assumed that we actually deal with a something like the uncapacitated (metric) facility location problem. This means that the costs for connecting facilities and customers are distances in a metric space. If in addition, the facility locations are not explicitly distinguished from the customer location, but should be rather selected in a first step, this models the problem of finding a minimum cost clustering in a metric space. Notice that our preceding arguments also provide  $\mathcal{FPT}$ -membership in this case.

*Rational weights.* In all versions discussed up to now, including the one introduced in the very beginning, one could also allow rational (or even “real-number”) weights bigger than or equal to one (modelling actual costs more realistically). Apart from numerical considerations, this would not change anything except for the Section dealing with the fast subset convolution; that approach would not work here.

*Cheap serves.* In view of our discussion it might be also an idea to allow for (some) services of zero cost. Irrespectively of whether or not zero costs for opening up facilities are allowed, the complexity picture now changes dramatically:

**Theorem 4.** FACILITY LOCATION, *parameterized by  $k$ , becomes  $W[2]$ -complete when zero weight services are allowed.*

*Proof.* (Sketch) Consider the special case when opening any facility costs one unit, and all services are for free. Then, the problem corresponds to selecting (at most)  $k$  of the facility to serve all customers. In more graph-theoretic form, this exactly corresponds to RED-BLUE DOMINATING SET, which in turn can be seen to be equivalent to HITTING SET [8]. Hence, any HITTING SET instance can be seen as a special instance of the variant of FACILITY LOCATION that we are considering. Since HITTING SET is well-known to be  $W[2]$ -hard, this property transfers to our problem at hand.

Conversely, membership in  $W[2]$  can be seen, e.g., by showing how to solve our problem with a short multi-tape nondeterministic Turing machine, see [4].  $\square$

A quick analysis of the proof of the previous theorem yields:

**Theorem 5.** FACILITY LOCATION, *parameterized by  $\ell$ , is  $W[2]$ -complete.*

## 4 Applications: The MDL principle

FACILITY LOCATION and its variants have numerous applications, even when only allowing integer costs. We shall now describe some of the less obvious ones that one way or the other apply the Minimum Description length (MDL) principle.

### 4.1 Automizing the production of XML documents

The web standard exchange format XML is (possibly first) described in: Extensible Markup Language (XML) 1.0, T. Bray, J. Paoli, and C. M. Sperberg-McQueen, 10 February 1998, available at <http://www.w3.org/TR/REC-xml>. In his notes on this standard (see [www.xml.com/axml/notes/Any1.html](http://www.xml.com/axml/notes/Any1.html)), T. Bray wrote, commenting on the construction of document type descriptors (DTD), a kind of context-free grammars used to specify syntactic characteristics of XML documents:

“Suppose you’re *given an existing well-formed XML document* and you want to build a DTD for it. One way to do this is as follows:

1. Make a list of all the element types that actually appear in the document, and *build a simple DTD* which declares each and every one of them as ANY. Now you’ve got a DTD (not a very useful one) and a valid document.
2. Pick one of the elements, and work out how it’s actually used in the document. Design a rule, and *replace the ANY declaration with a . . . content declaration*. This, of course, is the *tricky part*, particularly in a large document.
3. Repeat step 2, working through the elements one by one, until you have a useful DTD.”

Hence, various systems—usually called *DTD generators*—were designed to automatize the process of designing DTD’s, best from known examples. The main problem is that of *generalization*: when given a number of sample documents which should fit the envisaged DTD, at what “moment” and in which way is the DTD generator supposed to switch from a mode where it only produces a trivial DTD (this could be either a very specific one that can only parse the given samples or a very general one, as attempted by Bray with his ANY declaration proposal) to a mode where it actually tries to cleverly guess the syntactical structure in the given documents. Since most DTD generators internally work from “most specific” to “general,” this process of making “intelligent guesses” is known as *generalization*.

Garofalakis *et al.* proposed for this purpose the system XTRACT [11]. This DTD generator is based on the *Minimum Description Length (MDL) principle*, which can be viewed as a formalization of Occam’s razor. In actual fact, this principle in connection with grammar induction (DTD generators are a special case of this field) was discussed earlier by D. Conkley and I. H. Witten in [7].

The *length of a description* consists of two parts:

1. the length of the theory (in bits) and
2. the length of the data (in bits) when encoded with the help of the theory.

The system XTRACT uses MDL to evaluate regular expression hypotheses. This yields the combinatorial problem MDL-OPTIMAL-CODING:

**Given:** a set  $R = \{r_1, \dots, r_n\}$  of regular expressions (over the basic alphabet  $\Sigma$ ) and a set of strings  $S = \{s_1, \dots, s_m\} \subset \Sigma^+$ ,  $k \in \mathbb{N}$

**Question:** Is it possible to find a subset  $R'$  of  $R$  such that

$$\sum_{r \in R'} |c(r)| + \sum_{s \in S} |c(s|R')| \leq k \quad ?$$

The coding function  $c$  is described in detail in [11].

As an example (taken from [11]), consider the strings

$$S = \{ab, abab, ac, ad, bc, bd, bbd\}.$$

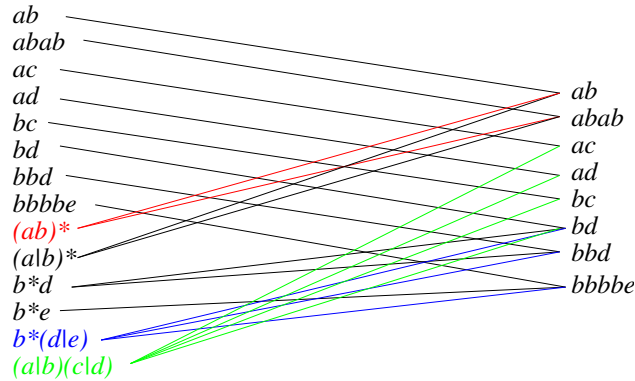
As hypotheses “covering” some of these examples, we consider, besides the elements from  $S$  itself, the following regular expressions

$$\{(ab)^*, b^*d, b^*e, b^*(d|e), (a|b)(c|d), (ab)^*\}.$$

Graphically, this covering relation can be depicted as a bipartite dominating set problem, see Fig. 1. To give an idea how the coding function  $c$  works, consider the three colored expressions. For example, to describe  $abab$  by the selected theory, i.e., by  $\{(ab)^*, b^*(d|e), (a|b)(c|d)\}$ , we need to describe that we take the first (out of the three expressions) and that we use *two* iterations. Likewise,  $bbbbe$  can be described by the second expression using *four* iterations.

Supplementing [11], it has been shown:

**Theorem 6.** (see [10]) MDL-OPTIMAL-CODING is  $\mathcal{NP}$ -complete.



**Fig. 1.** How to apply the MDL principle.

Garofalakis *et al.* propose using the related facility location problem, using the following translation: (a) place *facilities* (in our case: the selected theory) (b) into some *locations* (here: the regular expressions to choose from) (c) in order to minimize the *costs* incurred by the facilities (here: the number of bits needed to encode the theory) and by serving a given set of customers (in this case: the number of bits for encoding the given strings).

Our results imply that we can solve MDL-OPTIMAL-CODING as a parameterized problem to optimality. This is particularly important if MDL is actually used to *evaluate* hypotheses. Using here algorithms which only provide a logarithmic approximation guarantee will eventually mean that hypotheses will be rejected or preferred not according to their quality but according to the “quality” of the evaluation algorithm.

Since the number of bits (the costs) could be seen to be bounded by the input length itself, and all these numbers are integers, we can infer:

**Corollary 2.** MDL-OPTIMAL-CODING can be solved in time  $\mathcal{O}^*(2^n)$  (or also  $\mathcal{O}^*(2^k)$ ), where  $n$  denotes the number of input strings to be coded.

Observe that  $n < m$  in this particular application, since the input strings themselves are also always seen as possible regular expressions. So, the time bound derived by the preceding corollary is always superior to the trivial approach from Thm. 1.

Notice that similar methods are also used for data compression purposes. There, the “theory part” has sometimes to be explicitly encoded (as *side information*) and sometimes it is statically known to the (de)coder, which means that the “costs” of “opening up a facility” are zero. Hence, we face (again) the MEDIAN PROBLEM discussed in Sec. 3, so that this problem is also parameterized tractable, when parameterized by an upper bound on the number of bits used for the compressed data.

## 4.2 Computational Biology

Koivisto *et al.* described in [12] a method of applying ideas originating in the minimum description length principle to identify so-called *haplotype blocks* and to compare the strength of block boundaries. For our exposition, it is sufficient to know that intuitively, “a haplotype block can be considered to represent a sequence of ordered markers such that, for those markers, most of the haplotypes in the population cluster into a small number of classes. Each class consists of identical or almost identical haplotypes.” (Quoted from [12].)

They propose a dynamic programming algorithm for the problem of computing an optimal block structure and then to estimate the probabilities of each block boundary. This method relies on knowing a certain cost function whose computation actually is  $\mathcal{NP}$ -hard. However, this cost function (measuring the quality of the clusters/blocks) can be modeled with a  $k$ -MEANS PROBLEM and hence can be solved by the methods described in this paper. Since the involved weights grow at most exponential with the input length (as it appears to be generally true when dealing with weights in facility location problems that are derived from the MDL scenario), the fast subset convolution method can be used to further reduce the run times.

Again, there might be a broader connection to data compression; there, good  $k$ -means algorithms are crucial, e.g., in vector compression. The popular Lloyd algorithm (and variants thereof) may get stuck at local minima, so it might sometimes be a good idea to look for global maxima. Notice that exact algorithms might be a sensible approach here due to the slow convergence of Lloyd’s algorithm in practical situations, see [9].

## 5 Conclusions and Further Research

We have started a systematic study of facility location problems and variants. However, although this start looks quite promising, many things are still to be done. We sketch some of these in the following. (1) Even though many problems appear to be in  $\mathcal{FPT}$ , the algorithms we provided can be only seen as a starting point. Are there better algorithms for these important problems? (2) What about developing exact algorithms, possibly measured in  $N = n + m$ . Notice that the  $\mathcal{O}^*(2^n)$ - and  $\mathcal{O}^*(2^m)$ -algorithms shown in this paper can be easily abused in a WIN-WIN scenario to derive an  $\mathcal{O}^*(\sqrt{2}^N)$ -algorithm. (2) It is not quite clear if the basic assumption in parameterized algorithmics, namely, that the parameter is only moderately large, is met in this set of problems. Are there different parameterizations that are more suitable, at least in some applications? (3) Just to give two examples of a different, possibly additional parameter: (3a) In [6], a modified scenario was considered where some “outliers” were permitted, i.e., customers that could not be served (at decent costs). This could deliver a natural small parameter. (3b) In many situation, a company will already have opened a number of facilities, and the question is how to optimally improve the situation for the customers (and the company’s budget) by opening up a small number of new facilities.

Another aspect that has been neglected so far is geometry. Can we exploit metricity of costs to obtain better parameterized algorithms, as has been done in the case of approximation algorithms?

It is quite natural to assume that only (relatively) few facilities are within the “natural reach” of a single customer (it might be different from the viewpoint of the facilities, though). This implies a sort of degree-restriction on the side of the customer (in the underlying bipartite graph) which might yield a better runtime estimate of the dynamic programming algorithm, see [2].

## References

1. A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto. Fourier meets Möbius: fast subset convolution. In *Symposium on Theory of Computing STOC*, pages 67–74. ACM Press, 2007.
2. A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto. Trimmed Möbius inversion and graphs of bounded degree. In *Symposium on Theoretical Aspects of Computer Science STACS*, pages 85–96. IBFI, Schloss Dagstuhl, Germany, 2008.
3. A. Bumb. *Approximation algorithms for facility location problems*. PhD Thesis, Univ. Twente, The Netherlands, 2002.
4. M. Cesati. The Turing way to parameterized complexity. *Journal of Computer and System Sciences*, 67:654–685, 2003.
5. M. Charikar, S. Guha, E. Tardos, and D. S. Shmoys. A constant-factor approximation algorithm for the  $k$ -median problem. *Journal of Computer and System Sciences*, 65:129–149, 2002.
6. M. Charikar, S. Khuller, D. M. Mount, and G. Narasimhan. Algorithms for facility location problems with outliers. In *Symposium on Discrete Algorithms SODA*, pages 642–651. ACM Press, 2001.
7. D. Conklin and I. H. Witten. Complexity-based induction. *Machine Learning*, 16:203–225, 1994.
8. R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.
9. Q. Du, M. Emelianenko, and L. Ju. Convergence of the Lloyd algorithm for computing centroidal Voronoi tessellations. *SIAM Journal on Numerical Analysis*, 44:102–119, 2006.
10. H. Fernau. Extracting minimum length Document Type Definitions in NP-hard. In *Grammatical Inference: Algorithms and Applications; 7th International Colloquium ICGI*, volume 3264 of *LNCS/LNAI*, pages 277–278. Springer, 2004.
11. M. Garofalakis, A. Gionis, R. Rastogi, S. Seshadri, and K. Shim. XTRACT: learning document type descriptors from XML document collections. *Data Mining and Knowledge Discovery*, 7:23–56, 2003.
12. M. Koivisto, H. Manila, M. Perola, T. Varilo, W. Hennah, J. Ekelund, M. Lukk, L. Peltonen, and E. Ukkonen. An MDL method for finding haplotype blocks and for estimating the strength of block boundaries. In *Pacific Symposium on Biocomputing 2003 (PSB 2003)*, pages 502–513, 2002, see <http://psb.stanford.edu/psb-online/proceedings/psb03/>.
13. D. B. Shmoys, E. Tardos, and K. Aardal. Approximation algorithms for facility location problems (extended abstract). In *Symposium on Theory of Computing STOC*, pages 265–274. ACM Press, 1997.
14. N. E. Young.  $k$ -medians, facility location, and the Chernoff-Wald bound. In *Symposium on Discrete Algorithms SODA*, pages 86–95. ACM Press, 2000.