Embedding Complexity
A New Divide & Conquer Approach to Discrete Optimization,
Exemplified by the Steiner Tree Problem

D. Cieslik
Institute of Mathematics and Computer Science,
University of Greifswald,
D-17487 Greifswald, Germany,*

A. Dress‡
FSPM-Strukturbildungsprozesse,
University of Bielefeld,
D-33501 Bielefeld, Germany,

K. T. Huber‡
FMI, Mid Sweden University,
S 851-70 Sundsvall, Sweden,
and

V. Moulton§
FMI, Mid Sweden University,
S 851-70, Sundsvall, Sweden.

November 2, 2001
Abstract

In this note, we describe a new and quite natural way of analyzing instances of discrete optimization problems in terms of what we call the embedding complexity of an associated (more or less) canonical embedding of the (in general, vast) solution space of a given problem into a product of (in general, small) sets. This concept arises naturally within the context of a general Divide & Conquer strategy for solving discrete optimization problems using dynamic-programming procedures. As an example, an algorithm for solving Steiner’s problem on graphs with bounded treewidth is presented in detail.

Keywords: Optimization, discrete optimization, Divide & Conquer, dynamic programming, dynamic-programming schemes, algorithmic complexity, embedding complexity, Steiner’s problem, Steiner minimal trees, SMT, treewidth.

1 Introduction

One of the main challenges in optimization theory is to develop ways for understanding (a) what it is that makes a difficult problem difficult and (b) why one optimization problem can be solved easily whereas an apparently rather similar one is ‘hard’. Clearly, differentiating between polynomial (P) and non-deterministically polynomial (NP) problems has been groundbreaking in this context (see e.g. [14]). However, concepts like treewidth (see e.g [1, 2, 3, 4]) testify that, depending on which information is at hand, instances of ‘one and the same type’ of a problem can turn out to be either P...
or NP (notwithstanding the fact that we still do not know whether P ≠ NP holds); there just isn’t a unique canonical way to specify an instance of an optimization problem of given type.

In this note, we describe a new and, as we feel, quite natural way of analyzing instances of discrete optimization problems in terms of the embedding complexity of an associated (more or less) canonical embedding of the (in general, vast) ‘solution space’ of a given problem instance into a product of (in general, small) sets that is used to describe exactly those local features of a potential solution from which its total score can be computed additively.

A variety of basic examples (cf. Section 2.3) will be presented below to help clarify this simultaneously rather abstract, yet also very simple idea, including a detailed description of an application to Steiner’s problem. Two points in favor of this approach may, however, be mentioned immediately:

- It separates completely the numerical details of a given problem instance from its structural aspects.

- It allows to evaluate the efficiency of many heuristic procedures which, in the light of our approach, can be interpreted as methods that focus attention to a subspace of the total solution space containing (most probably) the optimal solution and having a much lower embedding complexity.

In this way, it is hoped to eventually develop a methodology for dealing with discrete optimization problems that can simultaneously produce (a) estimates for the work needed for solving any instance of a given class of problems exactly, and (b) for solving them heuristically.

In Section 2, we present a general scheme for analyzing discrete optimization problems. In the next two sections, we use this scheme to develop a Divide & Conquer approach for solving such problems — in this context, the concept of embedding complexity turns up naturally. Then, we present a rather general version of Steiner’s problem and, in the last section, we describe how the
general approach that we have developed before can be employed to deal with (our version of) Steiner’s problem for (hyper)graphs of bounded treewidth.

2 A General Scheme for “Linearizing” Discrete Optimization Problems

In this section, we present a scheme for rewriting a discrete optimization problem in such a way that the objective function can be viewed as an additive combination of local score functions. As will be shown in a subsequent paper, one of the various advantages of this approach is that it allows to explore the suitability of local-improvement approaches in a systematic fashion.

Our rewriting procedure can be applied to a wide range of discrete optimization problems, including all popular ones that we know of. To illustrate our procedure, we will apply it to three standard combinatorial-optimization problems.

2.1 Discrete Optimization Problems with an Additive Objective Function

Many optimization problems can be viewed as particular instances of the problem of deriving information regarding appropriate global states (or structures) from information regarding (1) the associated local states (or substructures) and (2) legitimate ways to fit these local data together so as to derive and to evaluate admissible global states (or structures). Consequently, a standard set-up to present such problems is the following one:

One starts with

(a) a family \((P_e)_{e \in E}\) of (hopefully small) sets \(P_e\) consisting of all local states to be taken into consideration at locality \(e\) where \(e\) runs through some index set \(E\) representing the set of relevant localities that we will as-
sume to be finite (though, more generally, it could also be assumed to be a measurable space),

(b) a family of maps \((s_e : P_e \to \overline{\mathbb{R}} := \mathbb{R} \cup \{\infty\})_{e \in E}\) associating a local score (or fitness value, penalty, or objective function) \(s_e(\pi_e) \in \overline{\mathbb{R}}\) to each local state \(\pi_e \in P_e\), and

(c) a subset \(R \subseteq \prod_{e \in E} P_e\) consisting of all globally admissible families of local states representing the set of all admissible global states \(\pi = (\pi_e)_{e \in E}\).

Given all that, the goal is to find some or all \(\pi = (\pi_e)_{e \in E} \in R \subseteq \prod_{e \in E} P_e\) that minimize the additive combination \(s(\pi) := \sum_{e \in E} s_e(\pi_e)\) of the local scores \(s_e(\pi_e), e \in E\). In other words, one defines the global objective function

\[ s : \prod_{e \in E} P_e \to \overline{\mathbb{R}} \]

simply as the sum by

\[ s(\pi) := \sum_{e \in E} s_e(\pi_e) \]

of the local scores \(s_e(\pi_e)\) of any \(\pi = (\pi_e)_{e \in E} \in \prod_{e \in E} P_e\), and one wants to find its minimum — however, not the minimum this function attains on the set \(\prod_{e \in E} P_e\) which would be easy (provided the minimum of the various local score functions \(s_e, e \in E\), can be computed easily one by one), but that which is attained on the subset \(R\), i.e. one aims to find the number

\[ s^* := \min(s(\pi) \mid \pi \in R). \]

In addition, one also wants to find the argument \(\arg\min[s(\pi) \mid \pi \in R]\) at which this minimum is attained, i.e. all (or, at least, one of the) elements \(\pi^* \in R\) with \(s(\pi^*) = s^*\). And whether this is difficult or not depends, as we shall see, on nothing but the way in which the set \(R\) can be described as a subset of the product set \(\prod_{e \in E} P_e\).

**Remark 2.1** Note that in the more general situation where \(E\) is a measurable space with measure \(\mu\), the family \((P_e)_{e \in E}\) needs to be replaced by
a fiber space $p : P \to E$ over $E$, the family of maps $(s_e)_{e \in E}$ by just one map $S : P \to \mathbb{R}$, and the product space $\prod_{e \in E} P_e$ by the set $\Gamma = \Gamma(p, S)$ of all sections $\pi : E \to P$ for which the resulting map $S \circ \pi : E \to \mathbb{R}$ is measurable, $R$ is considered as above to be a subset of $P$, and the map $s : \Gamma \to \mathbb{R}$ is defined quite naturally by $s(\pi) := \int_E (S \circ \pi)d\mu$.

### 2.2 The Rewriting Procedure

To rewrite a given optimization problem in the form described above, one proceeds as follows: Let $R'$ denote the space of all potential solutions and let $s' : R' \to \mathbb{R}$ denote the map whose minimum we wish to find. Look at the specific way in which $s'$ is defined and try to identify a family of feature maps $p_e : R' \to P_e$ indexed by some appropriately chosen set $E$; these maps are considered to associate, to each potential solution $r' \in R'$ and each feature $e \in E$, the specific instance $p_e(r')$ of that feature attained at $r'$. Moreover the feature maps have to be chosen in such a way that there exist maps $s_e : P_e \to \mathbb{R}$ ($e \in E$) so that, for every $r' \in R'$, the score $s'(r')$ coincides with the additive combination $\sum_{e \in E} s_e(p_e(r'))$ of the local scores $s_e(p_e(r'))$ of the images $p_e(r')$ of $r'$:

$$s'(r') := \sum_{e \in E} s_e(p_e(r')).$$

Replacing then $R'$ by its image $R := \{p_e(r) \mid r \in R'\}$ in $\prod_{e \in E} P_e$ (whether or not $r \neq r'$ implies the existence of some $e \in E$ with $p_e(r) \neq p_e(r')$), it is easy to see that solving the optimization problem for $R'$ and $s'$ is equivalent to solving the problem set up above for $(P_e)_{e \in E}$, $(s_e : P_e \to \mathbb{R})_{e \in E}$, and $R \subseteq \prod_{e \in E} P_e$.

### 2.3 Examples

To illustrate the above rewriting procedure, we present three standard optimization problems in a linearized form.
The **Traveling Salesman problem**: Here, one is given a set $V$ of cities and a weight function $d : \binom{V}{2} \to \mathbb{R}$ which assigns to every pair $\{x, y\} \in E := \binom{V}{2}$ the *distance* $d(x, y)$. One is then asked to find a tour (i.e. a Hamilton cycle in the graph $(V, E)$) of minimal length.

To *linearize* this problem in a form as described above, we put $P_e := \{0, 1\}$, $s_e(0) := 0$, and $s_e(1) := d(x, y)$ for every $e = \{x, y\} \in E$, and we define the subset $R$ of $\prod_{e \in E} P_e$ by

$$R := \{(\pi_e)_{e \in E} \in \prod_{e \in E} P_e \mid \{e \in E \mid \pi_e = 1\} \text{ is a tour}\}.$$ 

Then the task of finding the shortest tour is indeed identical to that of minimizing the function $s : \prod_{e \in E} P_e \to \mathbb{R}$ with $s((\pi_e)_{e \in E}) := \sum_{e \in E} s_e(\pi_e)$ over all elements in $R$.

The **pairwise alignment problem**: In biologically motivated sequence analysis, one is often confronted with the problem of finding, in some well-defined sense, the best *alignment* of a given set of DNA or amino-acid sequences. That is, given a family of such sequences (of various length), one has to insert dots “.”, indicating the so called *gaps*, at appropriate locations within the given sequences so that

- the elongated sequences are all of the same length,
- there is no position such that all elongated sequences have a gap at that position, and
- the elongated sequences are *as similar as possible* according to some predefined scoring scheme

(cf. [15, 21, 22] for details). In the case of a pair of sequences $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_m)$, this problem involves studying the set $E$ defined by

$$E := \{(i, \cdot), (\cdot, j), (i, j) \mid 1 \leq i \leq n, 1 \leq j \leq m\} = \left[\{1, \ldots, n\} \cup \{\cdot\}\right] \times \left[\{\cdot\} \cup \{1, \ldots, m\}\right] - \{\langle \cdot, \cdot \rangle\}.$$
For each $e \in E$, one puts $P_e := \{0, 1\}$ as above, and one defines the local scores $s_e : P_e \to \mathbb{R}$, by $s_e(0) := 0$ and

$$s_e(1) := \begin{cases} 
\text{“indel penalty”} & \text{if } e = (i, \cdot) \text{ or } e = (\cdot, j), \\
\text{“dissimilarity of } (i, j)\text{”} & \text{if } e = (i, j),
\end{cases}$$

where “indel penalty” and “dissimilarity of $(i, j)$” are some non-negative real numbers appropriately selected for the given task. And one defines $R$ to consist of all families $(\pi_e)_{e \in E} \in \prod_{e \in E} P_e$ for which the following holds:

(i) $1 \leq i \leq n \Rightarrow \pi_{(i, \cdot)} + \sum_{1 \leq j \leq m} \pi_{(i,j)} = 1$,

(ii) $1 \leq j \leq m \Rightarrow \pi_{(\cdot, j)} + \sum_{1 \leq i \leq n} \pi_{(i,j)} = 1$,

(iii) $1 \leq i < i' \leq n, 1 \leq j < j' \leq m \Rightarrow \pi_{(i,j)} + \pi_{(i',j')} \leq 1$.

Again, the task is to minimize the function $s : \prod_{e \in E} P_e \to \mathbb{R}$ over all elements in $R$.

The spin-glass problem: Our rewriting procedure was originally developed in [10, 9] to deal with the spin-glass problem, i.e. the problem of computing the minimum $\min(Q(x) \mid x \in \{\pm 1\}^n)$ of the values $Q(x)$ of a quadratic form $Q : \mathbb{R}^n \to \mathbb{R} : (x_1, \ldots, x_n) \mapsto \sum_{1 \leq i, j \leq n} c_{ij} x_i x_j$ that is attained at the vertices $x \in \{\pm 1\}^n$ of the hypercube $[-1, +1]^n \subseteq \mathbb{R}^n$. Here, one puts

$$E := \binom{\{1, \ldots, n\}}{2},$$

$$P_e := \{\pm 1\}^{|\{i, j\}|} (\{i, j\} \in E),$$

$$s_{\{i,j\}}(\pi_{\{i,j\}}) := (c_{ij} + c_{ji})\pi_{\{i,j\}}(i)\pi_{\{i,j\}}(j) (\{i, j\} \in E, \pi_{\{i,j\}} \in P_{\{i,j\}}),$$

and

$$R := \{(\pi_e)_{e \in E} \in \prod_{e \in E} P_e \mid \pi_e(i) = \pi_f(i) \text{ for all } i \in \{1, \ldots, n\}$$

and $e, f \in E$ with $i \in e \cap f$}. 

8
For details see [10] where it has been shown that, using this set-up, the minimum of $Q$ on $\{\pm1\}^n$ as well as the partition function of the spin-glass system described by $Q$ can be computed in $2^wO(n)$ steps where $w$ denotes the treewidth of the graph whose vertex set is the set of spins $\{1,\ldots,n\}$ and whose edge set consists of all pairs of interacting spins, i.e. all two-element subsets $\{i,j\}$ of $\{1,\ldots,n\}$ with $c_{ij} + c_{ji} \neq 0$. In [10], an application to maximum parsimony trees (cf. [11, 13, 15, 21, 22]) has also been discussed; more recently, it has been shown in [6] that our rewriting procedure can also be used to compute the entropy of the resulting probability distribution defined on the space $R = \{\pm1\}^n$ in $2^wO(n)$ steps.

These examples demonstrate that any discrete optimization problem can be rewritten in this way provided

- the elements in its search space can be described in terms of “local” data and
- its score can be computed as a sum of associated local scores.

### 3 A Divide & Conquer Approach

In this section, we develop a Divide & Conquer approach for solving optimization problems with an additive objective function as described in Section 2.1. We will also see that this leads to the concept of embedding complexity which permits to quantify a new feature of computational complexity associated with discrete optimization problems.

We begin by noting a simple fact: Suppose that we are given two sets $A_1$ and $A_2$ and two maps $f_1 : A_1 \rightarrow \mathbb{R}$ and $f_2 : A_2 \rightarrow \mathbb{R}$, and that we want to find the minimum value of $f : A_1 \times A_2 \rightarrow \mathbb{R}$: $f(a_1, a_2) \mapsto f_1(a_1) + f_2(a_2)$ attained at some subset $R$ of $A_1 \times A_2$. In general, this could require $\#R - 1$ comparisons. However, if $R$ can be written in the form $R = \bigcup_{j=1}^{k} A_{1j} \times A_{2j}$ where, for $1 \leq j \leq k$, the sets $A_{1j}$ and $A_{2j}$ are appropriately chosen subsets of $A_1$ and $A_2$, respectively, the number of comparisons required can be reduced...
dramatically. Namely, putting \( P := \{1, \ldots, k\} \) and defining \( R_i \subseteq P \times A_i \) for \( i = 1, 2 \) by

\[
R_i := \{(j, a_i) \mid j \in P \text{ and } a_i \in A_{ij}\},
\]
we see that the minimum value of \( f \) on

\[
R = R_1 \circ_P R_2 := \{(a_1, a_2) \mid \emptyset \neq \{j \in P \mid (j, a_1) \in R_1 \text{ and } (j, a_2) \in R_2\}\}
\]
is precisely the minimum value of the function \( g : P \to \mathbb{R} \) defined by

\[
g(j) := \min(f_1(a_1) \mid a_1 \in A_{1j}) + \min(f_2(a_2) \mid a_2 \in A_{2j}).
\]

Hence, the minimum value of \( f \) can be found in this case using

\[
(k - 1) + \sum_{j=1}^{k} (#A_{1j} - 1) + \sum_{j=1}^{k} (#A_{2j} - 1) = \sum_{j=1}^{k} (#A_{1j} + #A_{2j}) - k - 1
\]
comparisons.

Consequently, given a set-up as described in Section 2.2, one might start dealing with it by first searching for a bipartition \( E = F_1 \cup F_2 \) of \( E \) into two disjoint non-empty subsets \( F_1, F_2 \) of \( E \) such that the approach outlined above can be applied with respect to \( A_i := \prod_{e \in F_i} P_e \) \((i = 1, 2)\), i.e. such that a (comparatively) small set \( P \) as well as subsets

\[
R_i \subseteq P \times A_i \quad (i = 1, 2)
\]
can be found so that \( R \) coincides with the set \( R_1 \circ_P R_2 \) consisting of all global states

\[
(\pi_e)_{e \in E} \in \prod_{e \in E} P_e = A_1 \times A_2
\]
for which there exists some \( j \in P \) with

\[
(j, (\pi_e)_{e \in F_i}) \in R_i
\]
for \( i = 1, 2 \).
This would allow to reduce the original task to performing the same task several times, yet in the hopefully simpler situations where \( R \) is replaced by the subsets
\[
R_j^i := \{(\pi_e)_{e \in F_j} \mid (j, (\pi_e)_{e \in F_j}) \in R_i\} \quad (i = 1, 2, \ j \in P).
\]

In this way, the usual savings afforded by using a Divide & Conquer approach can be obtained within our set-up as described in Section 2.2. In the next section, we will show how this approach can further be elaborated recursively by considering whole hierarchies \( F \) of subsets of \( E \) (rather than just a bipartition of \( E \)) and using a dynamic-programming scheme for performing the required computations.

### 4 Dynamic-Programming Schemes

Given any set \( X \), recall that

- a collection \( F \) of subsets of \( X \) is called a hierarchy if \( F \) does not contain the empty set and \( F_1 \cap F_2 \in \{\emptyset, F_1, F_2\} \) holds for all \( F_1, F_2 \in F \);

- \( \#F \leq 2 \cdot \#X - 1 \) holds for any such hierarchy \( F \) (where \( \#A \) denotes the cardinality of a set \( A \)), and

- equality holds for a finite set \( X \) if and only if \( F \) is a maximal\(^1\) or, equivalently, a binary hierarchy, i.e. if there exist exactly two disjoint sets \( F_1, F_2 \in F \) for every \( F \in F^* := \{F \in F \mid \#F > 1\} \) for which \( F_1 \cup F_2 = F \) holds in which case we define \( E := \{F_1, F_2\} \).

Now, suppose that we are given the set \( E \) together with the family \( (P_e)_{e \in E} \) and the set \( R \subseteq \prod_{e \in E} P_e \), as introduced in Section 2.1, (a) and (c), as well as a maximal hierarchy \( F \) of subsets of the set \( E \). We identify each \( e \in E \) with the associated one-element subset \( \{e\} \in F \), allowing us to denote \( P_e \) also by \( P_{\{e\}} \) and to refer to an element of \( P_e = P_{\{e\}} \) also by \( \pi_{\{e\}} \) as well as by \( \pi_e \).

\(^1\)See [5] for a rather general discussion of maximal hierarchies.
We will say that a triple

\[ T := (\mathcal{F}, (P_F)_{F \in \mathcal{F}}, (R_F)_{F \in \mathcal{F}^*}) \]

consisting of

- a maximal hierarchy \( \mathcal{F} \) of subsets of \( E \),
- a family \( (P_F)_{F \in \mathcal{F}} \) of (appropriately chosen) sets \( P_F, F \in \mathcal{F} \) (with \( P_{\{e\}} := P_e \) as explained above), and
- a family \( (R_F)_{F \in \mathcal{F}^*} \) of (appropriately chosen) subsets \( R_F \subseteq P^F := P_F \times \prod_{F' \in E} P_{F'} \),

is a [dynamic-programming scheme for R](#) whenever an element \( \pi = (\pi_e)_{e \in E} \) of \( \prod_{e \in E} P_e \) is contained in \( R \) if and only if there exists a family \( (\pi_F)_{F \in \mathcal{F}} \) of elements \( \pi_F \in P_F \) (\( F \in \mathcal{F} \)) with \( \pi_{\{e\}} = \pi_e \) for all \( e \in E \) and \( (\pi_F, (\pi_{F'})_{F' \in E}) \in R_F \) for all \( F \in \mathcal{F}^* \). Clearly, defining the subset \( \overline{R} = \overline{R}(T) \) of \( \prod_{F \in \mathcal{F}} P_F \) by

\[
\overline{R} := \{ (\pi_F)_{F \in \mathcal{F}} \in \prod_{F \in \mathcal{F}} P_F \mid F \in \mathcal{F}^* \Rightarrow (\pi_F, (\pi_{F'})_{F' \in E}) \in R_F \},
\]

the condition that \( (\pi_F, (\pi_{F'})_{F' \in E}) \in R_F \) should hold for all \( F \in \mathcal{F}^* \) for a given family \( (\pi_F)_{F \in \mathcal{F}} \) is obviously equivalent to the condition that this family is contained in \( \overline{R} \).

Adapting the procedure from the previous section to this more general situation, we may now compute maps

\[ s_F : P_F \to \overline{R} \]

recursively for any \( F \in \mathcal{F}^* \) (including, of course, the set \( E \)), by putting

\[ s_F(\pi_F) := \min(\sum_{F' \in E} s_{F'}(\pi_{F'}) \mid \pi_{F'} \in P_{F'} \text{ and } (\pi_F, (\pi_{F'})_{F' \in E}) \in R_F), \]

allowing us to state the fundamental insight on which a Divide & Conquer approach using the above dynamic-programming scheme can be based, as follows:
Theorem 4.1 Given a dynamic-programming scheme \((\mathcal{F}, (P_F)_{F \in \mathcal{F}}, (R_F)_{F \in \mathcal{F}^*})\) for a subset \(R\) of a product set \(\prod_{e \in E} P_e\), the minimum \(\min(s_E(\pi_E) \mid \pi_E \in P_E)\) of the values of the map \(s_E : P_E \to \mathbb{R}\) computed recursively according to the equations (2) and (3), always coincides with the number

\[ s^* = \min(s(\pi) = \sum_{e \in E} s_e(\pi_e) \mid \pi \in R). \]

In addition, one, and even all, \(\pi \in R\) with \(s(\pi) = s^*\) can be found by working down through the hierarchy \(\mathcal{F}\) from any \(\pi_E \in P_E\) with \(s_E(\pi_E) = s^*\) and applying the argument function

\[ \arg\min(\sum_{F' \in \mathcal{F}^*} s_{F'}(\pi_{F'}) \mid \pi_{F'} \in P_{F'} \text{ and } (\pi_{F'}, (\pi_{F'})_{F' \in \mathcal{F}^*}) \in R_F)\]

to find, also recursively, one or all \(\pi_{F_1} \in R_{F_1}\) and \(\pi_{F_2} \in R_{F_2}\) for any \(\pi_{F} \in R_{F}\) one has encountered before.

We leave the simple proof to the reader. In view of this result, it makes sense to define the computational complexity of \(R\) relative to a given dynamic-programming scheme \((\mathcal{F}, (P_F)_{F \in \mathcal{F}}, (R_F)_{F \in \mathcal{F}^*})\) for \(R\) to be the sum (or, in view of \(\sum_{F \in \mathcal{F}^*} \#R_{F} = (\#E - 1)\#R_{\mathcal{F}^*} \leq (\#E - 1) \max(\#R_{F} \mid F \in \mathcal{F}^*), \) just the average \(\#R_{F}\) or the maximum) of the cardinalities of all the relations \(R_{F}\) occurring in the associated computational procedure, and to define the embedding complexity \(\text{embc}(R)\) of a subset \(R \subseteq \prod_{e \in E} P_e\) of a product set \(\prod_{e \in E} P_e\) to be the minimum of the complexities of \(R\) relative to any such scheme.

Yet, rather than following up these ideas any further here, discussing their relationship with the treewidth concept and similar measures of complexity or their use for computing also Boltzmann’s partition function

\[ f(k) := \sum_{\pi \in R} \exp(-ks(\pi)) \]
as well as the associated entropy
\[
\text{entr}(k) := \sum_{\pi \in R} -[\exp[-ks(\pi)]/f(k)] \ln[\exp[-ks(\pi)]/f(k)] = \sum_{\pi \in R} ks(\pi) \exp[-ks(\pi)]/f(k) + \ln(f(k))
\]
which will be done in separate papers (see also [6]), we will restrict our attention here now to showing that (and how) the above framework can be used in particular for the specific purpose of computing Steiner trees in weighted graphs of bounded treewidth.

**Remark 4.2** One might also consider dynamic-programming schemes of the form
\[
(\mathcal{F}, (P_F)_{F \in \mathcal{F}}, (R_{F,F'})_{F \in \mathcal{F}^*, F' \in \mathcal{F}})
\]
where — more or less as above — \(P_F\) and \(R_{F,F'} \subseteq P_F \times P_{F'}\) are appropriately chosen sets so that an element \(\pi = (\pi_e)_{e \in E} \in \prod_{e \in E} P_e\) is contained in \(R\) if and only if there exists a family \((\pi_F)_{F \in \mathcal{F}} \in \prod_{F \in \mathcal{F}} P_F\) of elements \(\pi_F \in P_F\) with \(\pi_{\{e\}} = \pi_e\) for all \(e \in E\) and \((\pi_F, \pi_{F'}) \in R_{F,F'}\) for all \(F\) in \(\mathcal{F}^*\) and \(F'\) in \(\mathcal{F}\). Indeed, the example discussed in the previous section fits even more easily into such a scheme rather than into the scheme described before. However, the scheme described first is more appropriate for the applications we will explain below while, in principle, both schemes can be transformed into each other without serious effort: Assume a dynamic-programming scheme \((\mathcal{F}, (P_F)_{F \in \mathcal{F}}, (R_F)_{F \in \mathcal{F}^*})\) for some subset \(R \subseteq \prod_{e \in E} P_e\) as described in (1) is given, and denote, for any \(\rho = (\pi_F, (\pi_{F'})_{F' \in \mathcal{F}'}) \in P_F^F\) \((F \in \mathcal{F}^*, F' \in \mathcal{F}')\), by \(\rho_F\) and \(\rho_{F'}\) the elements \(\pi_F\) and \(\pi_{F'}\), respectively. Then, the dynamic-programming scheme \((\mathcal{F}, (P'_F)_{F \in \mathcal{F}}, (R_{F,F'})_{F \in \mathcal{F}^*, F' \in \mathcal{F}})\) defined by
\[
P'_e := P_{\{e\}} = P_e
\]
for all \(e \in E\), and
\[
P'_F := R_F
\]
as well as
\[ R_{F,F'} := \{(\rho, \rho') \in P_F' \times P_{F'}' \mid \rho_{F'} = \rho_{F'} \} \]
for all \( F \in \mathcal{F}^* \) and \( F' \in \mathcal{F} \) (with \( \rho_{F'} := \pi_e \) in case \( F' = \{e\} \) and \( \rho' = \pi_e \), of course) is easily checked to be a dynamic-programming scheme for \( R \) of the form described in (4). Conversely, if such a scheme for \( R \) of the form (5) is given, the scheme \( (\mathcal{F}, (P_F)_{F \in \mathcal{F}}, (R_{F,F'})_{F \in \mathcal{F}^*}, (R'_{F})_{F \in \mathcal{F}^*}) \) defined by
\[ R'_F := \{ (\pi_F, (\pi_{F'})_{F' \in \mathcal{F}}) \in P_F \times \prod_{F' \in \mathcal{F}} P_{F'} \mid (\pi_F, \pi_{F'}) \in R_{F,F'} \text{ for every } F' \in \mathcal{F} \} \]
for every \( F \in \mathcal{F}^* \) is a scheme for \( R \) of the form considered originally.

5 The Steiner problem

Recall that, given any set \( X \), the set of all subsets of \( X \) — also called the power set of \( X \) — is denoted by \( \mathcal{P}(X) \), and the set of all non-empty subsets of \( X \) by \( \mathcal{P}^*(X) \).

We consider a finite (hyper)graph \( H = (V, E) \) specified in terms of a non-empty finite set \( V = V_H \), called its vertex set, and a collection \( E \subseteq \mathcal{P}^*(V) \) of non-empty subsets of \( V \), called its edge set. Given any subset \( A \subseteq V \) of \( V \) with \( A \subseteq \bigcup_{e \in E} e \) and any map \( s : E \to \mathbb{R}_{\geq 0} \), the Steiner problem relative to \( H, A \), and \( s \) is then to determine a subset \( F \subseteq E \) such that

- \( A \) is contained in the subset \( \bigcup F := \bigcup_{e \in F} e \) of \( V \),
- the set system \( F \) is relatively connected, i.e. \( F_1, F_2 \subseteq F, F = F_1 \cup F_2, \) and \( \bigcup F_1 \cap \bigcup F_2 = \emptyset \) implies \( F_1 = \emptyset \) or \( F_2 = \emptyset \), and
- \( s(F) := \sum_{f \in F} s(f) \) is as small as possible relative to the above two conditions.

Originally formulated by Hakimi [16] in 1971, Steiner’s problem for graphs has attracted considerable attention (reference books are [7, 17], an annotated
bibliography can be found in [20]). In particular, Steiner’s problem for graphs is well known to belong to the famous class of NP-complete problems [18]. Algorithms for solving the problem that are either exponential with respect to \#\(A\) and polynomial with respect to \#(\(V - A\)) or, conversely, polynomial with respect to \#\(A\) and exponential with respect to \#(\(V - A\)) can be found in [12, 19].

To rephrase Steiner’s problem so that it can be treated in the recursive fashion described in the previous sections, some conceptual effort is unfortunately unavoidable. More precisely, we have to consider the following more general problem: Suppose that instead of a single map \(s := E \to \mathbb{R}_{\geq 0}\) we are given a family of maps

\[ s_e : P_e := \mathcal{P}(\mathcal{P}^*(e)) \to \mathbb{R} \quad (e \in E) \]

that associate, to any collection \(\pi_e \subseteq \mathcal{P}^*(e)\) of non-empty subsets of \(e\) (including the empty collection), a certain weight \(s_e(\pi_e) \in \mathbb{R}\). Then, the general Steiner problem is to find a family \(\pi = (\pi_e)_{e \in E}\) of such collections \(\pi_e \subseteq \mathcal{P}^*(e)\) of non-empty subsets of \(e\) such that

(i) the set system \(\bigcup \pi = \bigcup_{e \in E} \pi_e \subseteq \mathcal{P}^*(V)\) is relatively connected, and

(ii) the sum \(s(\pi) = \sum_{e \in E} s_e(\pi_e)\) is as small as possible.

This problem contains Steiner’s problem for graphs as a special case: Consider as above a subset \(A \subseteq \bigcup E \subseteq V\) and a map \(s\) from \(E\) into \(\mathbb{R}_{\geq 0}\), assume — without loss of generality — that \(#e > 1\) holds for all \(e \in E\), and define the weight \(s'_e(\pi_e)\) for each \(e \in E\) and \(\pi_e \subseteq \mathcal{P}^*(e)\) by

\[ s'_e(\pi_e) := \begin{cases} 
  s(e) & \text{in case } \pi_e = \{e\}, \\
  0 & \text{in case } \pi_e = \{\{v\} \mid v \in A \cap e\}, \\
  \infty & \text{else}.
\end{cases} \]

Then, there is obviously a one-to-one correspondence between subsets \(F\) of \(E\) and families \(\pi = (\pi_e)_{e \in E} \in \prod_{e \in E} P_e\) with \(s'(\pi) = \sum_{e \in E} s'_e(\pi_e) < \infty\), given
by associating to any \( F \subseteq E \) the family \( \pi^F = (\pi^F_e)_{e \in E} \) defined by

\[
\pi^F_e := \begin{cases} 
\{e\} & \text{if } e \in F, \\
\{v\} \mid v \in A \cap e & \text{else}.
\end{cases}
\]

Moreover, we have

\[
s'(\pi^F) = \sum_{e \in E} s'(e) = \sum_{e \in E, \pi^F_e = \{e\}} s(e) = \sum_{f \in F} s(f) = s(F)
\]

for any such family \( \pi = \pi^F = (\pi^F_e)_{e \in E} \). Similarly, we have \( A \cap e \subseteq \bigcup \pi^F_e \) for each \( e \in E \) and, therefore,

\[
A = A \cap \bigcup_{e \in E} E \subseteq \bigcup_{e \in E} \bigcup \pi^F_e.
\]

Moreover, the set system \( \bigcup \pi^F = \bigcup_{e \in E} \pi^F_e \subseteq \mathcal{P}^*(V) \) is relatively connected if and only if the corresponding subset \( F \subseteq E \) is relatively connected and \( A \) is contained in \( \bigcup F = \bigcup_{e \in F} e \). Consequently, given the hypergraph \( (V, E) \), the solutions of the Steiner problem associated with \( A \subseteq V \) and \( s : E \to \mathbb{R}_{\geq 0} \) correspond in a one-to-one way to the solutions of the specific case of the more general problem we have specified above.

In the next section, we will explain how the general Steiner problem can be treated recursively in terms of an appropriate dynamic-programming scheme as discussed in Section 4.

### 6 The Divide & Conquer Approach for the General Steiner problem

We now explain how to solve the general Steiner problem by means of the Divide & Conquer approach discussed in Sections 3 and 4.

We begin with some definitions. Given a a finite hypergraph \( H = (V, E) \) and a set system \( F \subseteq \mathcal{P}^*(V) \), we denote the set all non-empty minimal elements in

\[
\{ A \subseteq \bigcup F \mid A \cap f \in \{\emptyset, f\} \text{ for all } f \in F \}
\]
by $\pi_0(F)$. We call the subsets of $V$ in $\pi_0(F)$ the connected components of the set system $F$ (or of the hypergraph $H$ in case $F = E$). In case $\pi_0(F) = \{V\}$ holds, the set system $F$ (or the hypergraph $H$ in case $F = E$) is also called a connected set system (or hypergraph, respectively). In case $\pi_0(F) = F$ holds for some set system $F \subseteq \mathcal{P}^*(V)$, we call $F$ a partial partition of $V$.

Now, assume that we are given

- a finite connected hypergraph $H = (V, E)$ with $e \neq \emptyset$ for all $e \in E$,
- a family of maps $s_e : P_e = \mathcal{P}(\mathcal{P}^*(e)) \rightarrow \mathbb{R}$ ($e \in E$) encoding local scores, and
- a binary hierarchy $\mathcal{F}$ of subsets of $E$.

Our goal is to compute $\min(s(\pi)) = \min(\sum_{e \in E} s_e(\pi_e))$ where $\pi = (\pi_e)_{e \in E}$ ranges over all families in

$$R(E) := \{\pi = (\pi_e)_{e \in E} \in \prod_{e \in E} P_e \mid \bigcup_{e \in E} \pi_e \text{ is relatively connected}\}.$$

In order to apply the Divide & Conquer strategy outlined in Section 3, we need to define the sets $P_F$ and $R_F$ for each $F \in \mathcal{F}^*$. We first define $P_F$:

Denote the set of all partial partitions of a subset $U$ of $V$ by $P_{\text{partial}}(U)$. For each subset $F$ of $E$ (whether in $\mathcal{F}$ or not), put

$$\partial F = \partial^E F := \bigcup F \cap \bigcup (E - F),$$

and $\partial e := \partial \{e\}$ for each $e \in E$. This allows us to define

$$P_F := P_{\text{partial}}(\partial F) \cup \{\bullet_F\}$$

for every $F$ in $\mathcal{F}^*$ where “$\bullet_F$” is an additional element not related to any of the elements in $V$ or $E$ which will be used to indicate that a family $\pi = (\pi_e)_{e \in E}$ of set systems $\pi_e \subseteq \mathcal{P}^*(e)$ has the property that the set system

$$\pi(F) := \bigcup_{e \in F} \pi_e$$
is non-empty and relatively connected while $\bigcup \pi(F) \cap \partial F = \emptyset$ holds. Note that $\partial F \neq \emptyset$ holds for all $F \subseteq E$ with $\emptyset \neq F \neq E$ in view of the connectedness of $H$ while $\partial E = \emptyset$ and, hence, $P_E = \{\emptyset, \bullet_E\}$ holds in case $F := E$.

Also, given a collection $\mathcal{A}$ of non-empty subsets of $V$ and a subset $U$ of $V$, put $\mathcal{A} \mid_U := \{A \cap U | A \in \mathcal{A} \text{ and } A \cap U \neq \emptyset\}$.

We now define the relation $R_F \subseteq P^F = P_F \times \prod_{F \in E} P_{F'}$ for any $F \in \mathcal{F}^*$ with, say, $E = \{F_1, F_2\}$ to consist of all $(\pi_F, \pi_{F_1}, \pi_{F_2}) \in P_F \times P_{F_1} \times P_{F_2}$ for which exactly one of the following assertions holds:

- **(R1)** $\pi_F = \bullet_F$, $F_1 \in \mathcal{F}^*$ and $\pi_{F_1} = \bullet_{F_1}$, $\pi_{F_2} = \emptyset$;
- **(R2)** $\pi_F = \bullet_F$, $\pi_{F_1} = \emptyset$, $F_2 \in \mathcal{F}^*$ and $\pi_{F_2} = \bullet_{F_2}$;
- **(R3)** $\pi_F = \bullet_F$, $\pi_{F_1} \neq \bullet_{F_1}$, $\pi_{F_2} \neq \bullet_{F_2}$, $\#\pi_0(\pi_{F_1} \cup \pi_{F_2}) = 1$, and $\partial F \cap (\pi_{F_1} \cup \pi_{F_2}) = \emptyset$;
- **(R4)** $\pi_F \neq \bullet_F$, $\pi_{F_1} \neq \bullet_{F_1}$, $\pi_{F_2} \neq \bullet_{F_2}$, $\pi_0(\pi_{F_1} \cup \pi_{F_2})|_{\emptyset F} = \pi_F$, and $\#\pi_0(\pi_{F_1} \cup \pi_{F_2}) = \#\pi_F$.

Using these definitions and notations, we can now state the main result of this paper:

**Theorem 6.1** The triple

$$T(\mathcal{F}) := (\mathcal{F}, (P_F)_{F \in \mathcal{F}}, (R_F)_{F \in \mathcal{F}^*})$$

is a dynamic-programming scheme for $R(E)$.

More precisely, the following holds: Given a family $\pi = (\pi_e)_{e \in E}$ in $\prod_{e \in E} P_e$, the corresponding set system $\pi(E) = \bigcup_{e \in E} \pi_e$ is the empty set system if and only if there exists a family $(\pi_F)_{F \in \mathcal{F}}$ in $\overline{R}(T(\mathcal{F})) \subseteq \prod_{F \in \mathcal{F}} P_F$ with $\pi_{\{e\}} = \pi_e$ for all $e \in E$ and $\pi_E = \emptyset$ while $\pi(E)$ is relatively connected and non-empty if and only if there exists a family $(\pi_F)_{F \in \mathcal{F}} \in \overline{R}(T(\mathcal{F}))$ with $\pi_{\{e\}} = \pi_e$ for all $e \in E$ and $\pi_E = \bullet_E$. 

19
Proof: Suppose that \((\pi_e)_{e \in E}\) is a family in \(\prod_{e \in E} P_e\). If \(\pi_0(E) = \emptyset\), then there exists indeed a family \((\pi_F)_{F \in \mathcal{F}} \in R(T(\mathcal{F}))\) with \(\pi_{(e)} = \pi_e\) for all \(e \in E\) and \(\pi_E = \emptyset\): In view of (R1), simply define \(\pi_F := \emptyset\) for all \(F \in \mathcal{F}\). Conversely if there exists such a family \((\pi_F)_{F \in \mathcal{F}} \in R(T(\mathcal{F}))\) with \(\pi_{(e)} = \pi_e\) for all \(e \in E\) and \(\pi_E = \emptyset\), then \(\pi_0(E) = \emptyset\) follows immediately from the definition of the relations \(R_F\) \((F \in \mathcal{F}^*)\) noting that \(F \in \mathcal{F}^*\), \(E = \{F_1, F_2\}\), and \(\pi_F = \emptyset\) implies \(\pi_{F_1} = \pi_{F_2} = \emptyset\) in view of (R4).

If there exists some family \((\pi_F)_{F \in \mathcal{F}}\) in \(R(T(\mathcal{F}))\) with \(\pi_{(e)} = \pi_e\) and \(\pi_E = \bullet_E\), then the definition of the relations \(R_F\) \((F \in \mathcal{F}^*)\) imply that there must exist a unique subset \(F_0 \in \mathcal{F}^*\) satisfying the following conditions:

1. \(\{F \in \mathcal{F}^* \mid \pi_F = \bullet_F\} = \mathcal{F}_{F_0 \subseteq} := \{F \in \mathcal{F} \mid F_0 \subseteq F\}\),
2. \(\pi_F = \emptyset\) for all \(F \in \mathcal{F}\) with \(F_0 \cap F = \emptyset\),
3. \(\#\pi_0(\bigcup_{F' \in F_0} \pi_{F'}) = 1\),
4. \(\partial F_0 \cap \bigcup_{F' \in F_0} \pi_{F'} = \emptyset\),
5. \(\pi_0(\bigcup_{F' \in F} \pi_{F'}) \big|_{\partial F} = \pi_F\) and \(\#\pi_0(\bigcup_{F' \in F} \pi_{F'}) = \#\pi_F\) holds for all \(F\) in \(\mathcal{F}_{\not\subseteq F_0} := \{F \in \mathcal{F}^* \mid F \not\subseteq F_0\}\).

We have to show that \(\#\pi_0(\pi(E)) = \#\pi_0(\pi(F_0)) = 1\) holds. To this end, we use the following fact that has been established in [8]: Given any family 
\[
\psi = (\psi_F)_{F \in \mathcal{F} \subseteq F_0} \in \prod_{F \in \mathcal{F} \subseteq F_0} P_F
\]
with \(\pi_0((\bigcup_{F' \in F} \psi_{F'}) \big|_{\partial F}) = \psi_F\) for all \(F \in \mathcal{F}_{\subseteq F_0}\) (and \(\mathcal{F} \subseteq F_0 := \{F \in \mathcal{F} \mid F \subseteq F_0\}\), of course), the identity
\[
\#\pi_0\left(\bigcup_{e \in F_0} \psi_{(e)}\right) - \#\psi_0 = \sum_{F \in \mathcal{F}_{\not\subseteq F_0}} \left[\#\pi_0(\bigcup_{F' \in F} \psi_{F'}) - \#\psi_F\right]
\]
always holds. Applying this identity to the family 
\[
\psi = (\psi_F)_{F \in \mathcal{F} \subseteq F_0}
\]
defined by
\[
\psi_F := \begin{cases} 
\pi_F & \text{if } F \neq F_0, \\
\emptyset & \text{if } F = F_0,
\end{cases}
\]
we have
\[
\psi = (\psi_F)_{F \in \mathcal{F} \subseteq F_0}.
\]
our claim follows readily: Indeed, the above identity together with (2) and (5) implies immediately that

\[
\# \pi_0(\pi(E)) = \# \pi_0(\pi(F_0)) = \# \pi_0(\bigcup_{e \in F_0} \psi(E)) = \# \pi_0(\bigcup_{e \in F_0} \psi(e)) - \# \psi_{F_0}
\]

\[
= \sum_{F \in \mathcal{F}^*_{F_0}} \left[ \# \pi_0(\bigcup_{F' \in \mathcal{E}} \psi_{F'}) - \# \psi_F \right] = \# \pi_0(\bigcup_{F' \in \mathcal{E}} \pi_{F'})
\]

\[
+ \sum_{F \in \mathcal{F}^*_{F_0}} \left[ \# \pi_0(\bigcup_{F' \in \mathcal{E}} \pi_{F'}) - \# \pi_F \right] = 1 + \sum_{F \in \mathcal{F}^*_{F_0}} 0 = 1
\]

must hold, exactly as claimed.

Conversely, assume that \# \pi_0(\pi(E)) = 1 holds. We have to find a family \((\pi_F)_{F \in \mathcal{F}} \in \mathcal{R}(T(\mathcal{F}))\) with \(\pi_{\{e\}} = \pi_e\) for all \(e \in E\) and \(\pi_E = \bullet_E\). To this end, let us first recall from [8] that, given any family \(\psi = (\psi_e)_{e \in E}\) in \(\prod_{e \in E} P_e\), the associated canonical extension \(\psi^* = (\psi^*_F)_{F \in \mathcal{F}} \in \prod_{F \in \mathcal{F}} P_F\) defined by \(\psi^*_F : \psi_e\) for every \(e \in E\) and \(\psi^*_F := \pi_0(\bigcup_{e \in F} \psi_e)\) for every \(F \in \mathcal{F}^*\), satisfies the identity

\[
\psi^*_F = \pi_0(\bigcup_{F' \in \mathcal{E}} \psi^*_{F'}) \big|_{\partial F}
\]

for every \(F \in \mathcal{F}^*\) which in turn, using the main result from [8] quoted already above, implies that

\[
\# \pi_0(\bigcup_{e \in E} \psi^*_e) = \sum_{F \in \mathcal{F}^*} \left[ \# \pi_0(\bigcup_{F' \in \mathcal{E}} \psi^*_{F'}) - \# \psi^*_F \right]
\]

must hold for any such family \(\psi\) and its extension \(\psi^*\). Consequently, applying this fact to the family \(\pi^* = (\pi^*_F) \in \prod_{F \in \mathcal{F}^*} P_F\) defined by \(\pi^*_F := \pi_0(\pi(F))\big|_{\partial F}\) for each \(F \in \mathcal{F}^*\) and \(\pi^*_{\{e\}} := \pi_e\) for each \(e \in E\), our assumption \# \pi_0(\pi(E)) = 1\) together with the obvious fact that

\[
\# \pi_0(\bigcup_{F' \in \mathcal{E}} \pi^*_{F'}) = \# \pi_0(\bigcup_{F' \in \mathcal{E}} \psi^*_{F'}) \big|_{\partial F} = \# \pi^*_F
\]

must hold for every \(F \in \mathcal{F}^*\), it follows that there must exist a unique \(F_0 \in \mathcal{F}^*\) with

\[
\# \pi_0(\bigcup_{F' \in F_0} \pi^*_{F'}) = 1 + \# \pi^*_0
\]
while
\[ \#\pi_0(\bigcup_{F' \in \mathcal{F}} \pi_{F'}^*) = \#\pi_F^* \]
must hold for all other \( F \in \mathcal{F}^* \). In view of \( \#\pi_E^* = 0 \), this implies immediately by induction with respect to \( \#\mathcal{F}_{F \subseteq} = \#\{F' \in \mathcal{F} \mid F \subseteq F'\} \) that \( \pi_F^* = \emptyset \) must hold for all \( F \in \mathcal{F} \) that are not properly contained in \( F_0 \). In particular, \( \pi_e = \emptyset \) must hold for every \( e \in E - F_0 \), and we must have
\[ \#\pi_0(\bigcup_{F' \in F_0} \pi_{F'}^*) = 1. \]
Consequently, the family \( (\pi_F)_{F \in \mathcal{F}} \in \prod_{F \in \mathcal{F}} P_F \) defined by
\[ \pi_F := \begin{cases} \bullet_F & \text{if } F_0 \subseteq F, \\ \pi_F^* & \text{else} \end{cases} \]
is an extension of \( \pi \) which is contained in \( R(T(\mathcal{F})) \) as required.

Remark 6.2 More generally, the results in [8] quoted above imply that, given any subset \( U \subseteq V \) and family \( \pi = (\pi_e)_{e \in E} \in \prod_{e \in E} P_e \), the obviously unique family
\[ (\pi_U^*, k_U^*)_{F \in \mathcal{F}} \in \prod_{\{e\} \in \mathcal{F}} (P^*(e) \times \{0\}) \times \prod_{F \in \mathcal{F}^*} (P_{\text{partial}}(U \cup \partial F) \times \mathbb{N}_0) \]
defined recursively by
(U1) \( \pi_{\{e\}}^U = \pi_e \) (and \( k_{\{e\}}^U = 0 \)) for every \( e \in E \)
and
(U2) \( \pi_F^U = \pi_0(\bigcup_{F' \in F} \pi_{F'}^U) \mid_{U \cup \partial F} \) and \( k_F^U = \#\pi_0(\bigcup_{F' \in F} \pi_{F'}^U) - \#\pi_F^U + \sum_{F' \in F} k_{F'}^U \)
for every \( F \in \mathcal{F}^* \).
satisfies the identities

$$\#\pi_0(\bigcup_{e \in E} \pi_e) = \#\pi_E^U + k_E^U$$

and

$$\pi_E^U = \pi_0(\bigcup_{e \in E} \pi_e) \big|_U.$$

Thus, we have

$$\#\pi_0(\bigcup_{e \in E} \pi_e) \leq 1 \iff k_E^0 \leq 1$$

$$\iff k_F^0 \leq 1 \text{ and } \pi_F^0 = \emptyset \text{ in case } k_F^0 = 1 \text{ for all } F \in \mathcal{F}$$

which in turn holds if and only if the family \((\pi_F)_{F \in \mathcal{F}}\) defined by

$$\pi_F := \begin{cases} 
\pi_F^0 & \text{if } k_F^0 = 0, \\
\bullet_F & \text{if } k_F^0 \neq 0
\end{cases}$$

is contained in \(\mathcal{R}(T(\mathcal{F}))\), while the existence of a family \((\pi_F)_{F \in \mathcal{F}} \in \mathcal{R}(T(\mathcal{F}))\)

with \(\pi_{(e)} = \pi_e\) for all \(e \in E\) implies conversely that the family \((\pi_F^0, k_F^0)_{F \in \mathcal{F}}\)

satisfies the equation

$$\left(\pi_F^0, k_F^0\right) = \begin{cases} 
(\pi_F, 0) & \text{if } \pi_F \neq \bullet_F, \\
(0, 1) & \text{if } \pi_F = \bullet_F
\end{cases}$$

for every \(F \in \mathcal{F}\). Clearly, this provides another proof of Theorem 6.1. More generally, we can use the families \((\pi_F^U, k_F^U)_{F \in \mathcal{F}}\) as constructed above to design a Divide & Conquer algorithm to search for families \((\pi_e)_{e \in E} \in \prod P_e\) that minimize (or maximize) the sum \(\sum_{e \in E} s_e(\pi_e)\) while \(\#\pi_0(\bigcup_{e \in E} \pi_e) \leq k\) (or \(= k\)) holds for some fixed \(k\) and \(\pi_0(\bigcup_{e \in E} \pi_e) \big|_U\) coincides with one of a pregiven collection of partial partitions of \(U\).

In particular, within the standard context of Steiner minimal trees described in Section 5, we can use our Divide & Conquer approach to determine “Steiner minimal \(k\)-forests” for any \(k \in \mathbb{N}\), i.e. subsets \(F\) contained in the set \(E\) of edges of a given graph \(G = (V, E)\) with vertex set \(V\) such that a
given subset $A$ of $V$ is contained in $\cup F = \bigcup_{e \in F} e$, $\#\pi_0(\cup F) = k$ holds, and $s(F) = \sum_{e \in F} s(e)$ is as small as possible relative to these two conditions for some pregiven positive weight function $s : E \to \mathbb{R}_{>0}$.

**Remark 6.3** Defining the *star-treewidth* $tw^*(H)$ of a finite hypergraph $H = (V, E)$ as the minimal number $k \in \mathbb{N}$ for which there exists some binary hierarchy $\mathcal{F} \subseteq \mathcal{P}(E)$ with $\#\partial(F) \leq k$ for all $F \in \mathcal{F}$, then except for the computation of the maps

$$s_e : P_e \to \mathbb{R}$$

which involves $P_e = \mathcal{P}(\mathcal{P}^*(e))$ for each $e \in E$ – the number of computational steps necessary for computing optimal families

$$(\pi_e)_{e \in E} \in \prod_{e \in E} P_e$$

for any given weighting scheme $(s_e : P_e \to \mathbb{R})_{e \in E}$ is bounded essentially by

$$(\#E - 1) \cdot p_{\text{partial}}(tw^*(H))^3,$$

with $p_{\text{partial}}(k)$ denoting $\#P_{\text{partial}}(X)$ for a set $X$ of cardinality $k$.

In a forthcoming paper, we will discuss the relation between (the standard concept of) treewidth, the star-treewidth, and the embedding complexity of subsets of product sets as defined in Section 4.

## References


