

# Heuristics for the data arrangement problem on regular trees

Eranda Çela · Rostislav Staněk

the date of receipt and acceptance should be inserted later

**Abstract** The data arrangement problem on regular trees (DAPT) consists in assigning the vertices of a given graph  $G$  to the leaves of a  $d$ -regular tree  $T$  such that the sum of the pairwise distances of all pairs of leaves in  $T$  which correspond to edges of  $G$  is minimised. This problem is a special case of the generic graph embedding problem (GEP) and is  $NP$ -hard for every fixed  $d \geq 2$ .

In this paper we propose construction and local search heuristics for the DAPT and introduce a lower bound for this problem. The analysis of the performance of the heuristics is based on two considerations: a) the quality of the solutions produced by the heuristics as compared to the respective lower bound b) for a special class of instances with known optimal solution we evaluate the gap between the optimal value of the objective function and the objective function value attained by the heuristic solution, respectively.

**Keywords** combinatorial optimisation · data arrangement problem · regular trees · heuristics

**Mathematics Subject Classification (2000)** 90C27

---

Eranda Çela  
Institut für Optimierung und Diskrete Mathematik, TU Graz, Steyrergasse 30, A-8010 Graz,  
Austria  
Tel.: +43-316-873105366,  
E-mail: cela@opt.math.tu-graz.ac.at

Rostislav Staněk  
Institut für Statistik und Operations Research, Universität Graz, Universitätsstraße 15,  
Bauteil E/III, A-8010 Graz, Austria  
Tel.: +43-316-3807246,  
E-mail: rostislav.stanek@uni-graz.at

## 1 Introduction

Given an undirected graph  $G = (V(G), E(G))$  with  $|V(G)| = n$ , an undirected graph  $H = (V(H), E(H))$  with  $|V(H)| \geq n$  and some subset  $B$  of the vertex set of  $H$ ,  $B \subseteq V(H)$ , with  $|B| \geq n$ , the *generic graph embedding problem* (GEP) consists of finding an injective embedding of the vertices of  $G$  into the vertices in  $B$  such that some prespecified objective function is minimised. Throughout this paper we will call  $G$  *the guest graph* and  $H$  *the host graph*. A commonly used objective function maps an embedding  $\phi: V(G) \rightarrow B$  to

$$\sum_{(i,j) \in E(G)} d(\phi(i)\phi(j)),$$

where  $d(x, y)$  denotes the length of the shortest path between  $x$  and  $y$  in  $H$ . The host graph  $H$  may be a weighted or a non-weighted graph; in the second cases the path lengths coincide with the respective number of edges. Given a non-negative number  $A \in \mathbb{R}$ , the decision version of the *GEP* asks whether there is an injective embedding  $\phi: V(G) \rightarrow B$  such that the objective function does not exceed  $A$ .

Different versions of GEP have been studied in the literature; the *linear arrangement problem*, where the guest graph is a path with  $n$  vertices, see Chung (1984), Juvan and Mohar (1992) and Shiloach (1979), is probably the most prominent among them. A number of other classical and well known combinatorial optimisation problems can be seen as special cases of the GEP, as e.g. the Hamiltonian cycle problem, the Hamiltonian path problem and the graph isomorphism problem (see e.g. ela and Staněk (2013) for a more detailed discussion of the relationship between these problems).

This paper deals with the version of the GEP where the guest graph  $G$  has  $n$  vertices, the host graph  $H$  is a complete  $d$ -regular tree of height  $\lceil \log_d n \rceil$  and the set  $B$  consists of the leaves of  $H$ .

**Definition 1** (*d-regular tree*) A tree  $T = (V, E)$  is called a *d-regular tree*,  $d \in \mathbb{N}$ ,  $d \geq 2$ , if

1. it contains a specific node  $v_0 \in V$  of degree  $d$  which is called the *root* of  $T$ ,
2. every vertex but the leaves and the root has degree  $d + 1$  and
3. there is a number  $h \in \mathbb{N}$  such that the length  $d(l, v_0)$  of the path between the root  $v_0$  and a leaf  $l$  equals  $h$  for every leaf  $l$  of  $T$ .

The number  $h$  is called *height* of the tree. For every vertex  $v \in V \setminus \{v_0\}$ , i.e. for any vertex  $v$  but the root, the unique neighbour of  $v$  in the path between  $v_0$  and  $v$  in  $T$  is called the *father* of  $v$ . All other neighbours of  $v$  (if any) are called the *children* of  $v$ . The neighbours of the root  $v_0$  are called *children* of  $v_0$ . The *level* of a vertex  $v$ , denoted by  $\text{level}(v)$ , is the length (i.e. the number of edges) of the unique path joining  $v$  and the root of the tree. Thus in a  $d$ -regular tree of height  $h$  the level of each leaf equals  $h$ , whereas the level of the root equals 0. All vertices  $w$ ,  $w \neq v$ , of the unique path joining  $v$  and the

root of the tree are called *ancestors of  $v$* . Given two vertices  $v$  and  $u$  their *most recent common ancestor  $w$*  is their common ancestor with the highest level, i.e.  $w = \operatorname{argmax}\{\operatorname{level}(t) : t \text{ is a common ancestor of } u \text{ and } v\}$ .

From now on we will denote the host graph by  $T$ . The height of  $T$  as specified above guarantees that the number  $|B|$  of leaves fulfills  $|B| \geq n$  and that the number of the predecessors of the leaves in  $T$  is smaller than  $n$ . Thus  $\lceil \log_d n \rceil$  is the smallest height of a  $d$ -regular tree which is able to accommodate an injective embedding of the vertices of the guest graph on its leaves. This problem is originally motivated by real problems in communication systems and was first posed by Luzcak and Noble (1992). We will call this version of the GEP *the data arrangement problem on regular trees (DAPT)*. Luzcak and Noble (1992) have shown that the DAPT is *NP*-hard for every fixed  $d \geq 2$ . The question about the computational complexity of the DAPT in the case where the guest graph is a tree, posed in Luzcak and Noble (1992), is still open. In this perspective the development of heuristic approaches to efficiently find good solutions to DAPT is a natural task. There are plenty of heuristics for different versions of the GEP in the literature, especially for the linear arrangement problem, see e.g. the papers by Petit (1998) and Petit (2003) for nice and comprehensive reviews. However, to our knowledge there are no specific heuristic approaches to solve the DAPT and no benchmark instances have been developed for this problem yet. In this paper we make a first step in this direction and propose construction and local search approaches as well as a lower bound for the DAPT, much in the spirit of Petit (1998) and Petit (2003) which deal with the linear arrangement problem. In order to evaluate the performance of the proposed heuristics we generate a number of families of test instances some of them being polynomially solvable or having a known optimal objective function value.

The paper is organised as follows. Section 2 discusses some general properties of the problem and introduces the notation used throughout the paper. In Section 3 we derive a lower bound for optimal objective function value to be used in the evaluation of the performance of solution heuristics. Section 4 introduces the proposed *heuristics*. Sections 5, 6 and 7 discuss the test instances, the numerical results and some conclusions and outlook, respectively.

## 2 Notation and general properties of the DAPT

Consider a guest graph  $G = (V, E)$  with  $n$  vertices,  $|V| = n$ , and a host graph  $T$  which is a  $d$ -regular tree of height  $h$ ,  $h := \lceil \log_d n \rceil$ . Let  $B$  be the set of leaves of  $T$ . Notice that due to the above choice of  $h$  we get the following upper bound for the number  $b = |B|$  of leaves:

$$b := |B| = d^h = d^{h-1}d < nd. \quad (1)$$

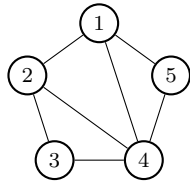
**Definition 2** An *arrangement* is an injective mapping  $\phi : V \rightarrow B$ . The *data arrangement problem on regular trees (DAPT)* asks for an *arrangement*  $\phi$  that

minimises the *objective value*  $OV(G, d, \phi)$

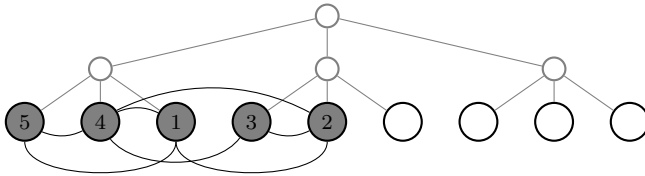
$$OV(G, d, \phi) := \sum_{(u,v) \in E} d_T(\phi(u), \phi(v)), \quad (2)$$

where  $d_T(\phi(u), \phi(v))$  denotes the length of the  $\phi(u)$ - $\phi(v)$ -path in the  $d$ -regular tree  $T$ . Such an arrangement is called an *optimal arrangement*. An instance of the DAPT is fully determined by the guest graph and the parameter  $d$  of the regular tree  $T$  which serves as host graph. Such an instance of the problem will be denoted by  $DAPT(G, d)$ .

Fig. 1 shows a guest graph  $G$  with vertices  $\{v_1, v_2, v_3, v_4, v_5\}$  and Fig. 2 shows a 3-regular tree of height  $2 = \lceil \log_3 5 \rceil$  as a host graph together with a minimum arrangement. The numbers in the leaves of  $T$  denote the vertex indices mapped to the leaves, respectively,



**Fig. 1** A guest graph.



**Fig. 2** An optimal arrangement which objective value is 20. The dark connecting arcs represent the edges of the guest graph. This holds for all pictures representing arrangements throughout this paper.

**Notation.** From now on let the set of vertices of the guest graph  $G$  be given as  $V(G) = \{v_1, \dots, v_n\}$  and let  $m := |E(G)|$  be its number of edges. We denote the *set of neighbours* of any vertex  $v$  by  $\Gamma(v)$ . We denote by  $h(T)$  the height of a  $(d)$ -regular tree  $T$ . A *basic subtree*  $T'$  of the  $d$ -regular tree  $T$  is a  $d$ -regular subtree of  $T$  with  $h(T') = h(T) - 1$  rooted at some child of the root of  $T$ . We consider a so-called *canonical order* of the  $2^h$  leaves of a  $d$ -regular tree of height  $h$  and denote the leaves by  $b_1, b_2, \dots, b_{d^h}$ , where the natural order of the subscripts corresponds to the canonical order of the leaves. The canonical

order of the leaves is defined recursively as follows. If  $h = 1$  the canonical order is an arbitrary but fixed order of the leaves. If  $h > 1$ , the children of the root are sorted arbitrarily as  $ch_1, ch_2, \dots, ch_d$ , and the leaves of the basic subtree  $T_i$  rooted at  $ch_i$ , for all  $1 \leq i \leq d$ , are sorted with respect to the canonical order in  $T_i$ . The canonical order of the leaves in  $T$  is then the unique order specified by the indices of the leaves in  $b_1, b_2, \dots, b_{d^h}$  such that  $b_{(i-1)d^{h-1}+1}, b_{(i-1)d^{h-1}+2}, \dots, b_{i \cdot d^{h-1}}$  are the leaves of the basic subtree  $T_i$  and the order of the subscripts corresponds to the canonical order of the leaves in  $T_i$ .

If the leaves are labelled according to the canonical order as above then the pairwise distances between the leaves of a  $d$ -regular tree are given by a simple formula.

**Observation 1** *Let  $T$  be a  $d$ -regular tree of height  $h$  and let its leaves be labelled according to the canonical ordering. The distances between the leaves in  $T$  are given as  $d_T(b_t, b_j) = 2l$ , where*

$$l := \min \left\{ k \in \{1, 2, \dots, h\} : \left\lfloor \frac{t-1}{d^k} \right\rfloor = \left\lfloor \frac{j-1}{d^k} \right\rfloor \right\}, \quad (3)$$

for all leaves  $b_t, b_j$  of  $T$  with  $t, j \in \{1, 2, \dots, d^h\}$ . If vertex  $u$  is the most recent common ancestor of  $b_t$  and  $b_j$ , then  $h - l$  equals the level of  $u$ .

*Proof* First let us observe that for all  $t, j \in \{1, 2, \dots, d^h\}$ ,  $\lfloor \frac{t-1}{d^l} \rfloor = \lfloor \frac{j-1}{d^l} \rfloor$  implies  $\lfloor \frac{t-1}{d^{l+1}} \rfloor = \lfloor \frac{j-1}{d^{l+1}} \rfloor$ , for all  $l \in \{1, 2, \dots, h-1\}$ .

We prove the claim by induction on  $h$ . If  $h = 1$  then  $T$  has  $d$  leaves labelled by  $1, 2, \dots, d$ , their pairwise distances are all equal to 2 and  $\lfloor \frac{t-1}{d} \rfloor = \lfloor \frac{j-1}{d} \rfloor = 0$ , so the claim holds. Assume that the claim holds for regular trees of height up to  $h-1$ . Consider now a tree of height  $h$  with leaves labelled by  $b_1, b_2, \dots, b_{d^h}$  in the canonical ordering and let  $b_t, b_j$  be two leaves of it.

Let  $t = (i_t - 1)d^{h-1} + r_t$  and  $j = (i_j - 1)d^{h-1} + r_j$  with  $i_t, i_j \in \{1, 2, \dots, d\}$  and  $r_t, r_j \in \{1, 2, \dots, d^{h-1}\}$ . Clearly  $\lfloor \frac{t-1}{d^{h-1}} \rfloor = i_t - 1$  and  $\lfloor \frac{j-1}{d^{h-1}} \rfloor = i_j - 1$ . Thus, if  $l := \min\{k \in \{1, 2, \dots, h\} : \lfloor \frac{t-1}{d^k} \rfloor = \lfloor \frac{j-1}{d^k} \rfloor\} = h$ , if and only if  $i_t \neq i_j$ , or equivalently,  $b_t, b_j$  are leaves of different basic subtrees of  $T$ . For leaves of different basic subtrees we have  $d_T(b_i, b_j) = 2h$  and hence, the claim holds in this case.

Otherwise  $l := \min\{k \in \{1, 2, \dots, h\} : \lfloor \frac{t-1}{d^k} \rfloor = \lfloor \frac{j-1}{d^k} \rfloor\} \leq h-1$  which implies  $\lfloor \frac{t-1}{d^{h-1}} \rfloor = \lfloor \frac{j-1}{d^{h-1}} \rfloor$ . Thus  $b_t$  and  $b_j$  are leaves of the same basic subtree of  $T$ . Let this be the  $r$ -th basic subtree  $T_r$  of  $T$  with leaves  $b_{(r-1)d^{h-1}+s}$  with  $s = 1, 2, \dots, d^{h-1}$ . In the canonical ordering in  $T_r$  these leaves would be labelled by  $b_s$ , for  $s = 1, 2, \dots, d^{h-1}$ . Let  $t = (r-1)d^{h-1} + s_t$  and  $j = (r-1)d^{h-1} + s_j$  for  $s_t, s_j \in \{1, 2, \dots, d^{h-1}\}$ .  $T_r$  is a  $d$ -regular tree of height  $h-1$  and hence  $d_{T_r}(b_{s_t}, b_{s_j}) = 2l$  holds, where  $l := \min\{k \in \{1, 2, \dots, h-1\} : \lfloor \frac{s_t-1}{d^k} \rfloor = \lfloor \frac{s_j-1}{d^k} \rfloor\}$ , according to our inductive assumption.

Finally notice that  $d_{T_r}(b_{s_t}, b_{s_j}) = d_T(b_t, b_j)$  and

$$\begin{aligned} l &:= \min \left\{ k \in \{1, 2, \dots, h-1\} : \left\lfloor \frac{s_t-1}{d^k} \right\rfloor = \left\lfloor \frac{s_j-1}{d^k} \right\rfloor \right\} \\ &= \min \left\{ k \in \{1, 2, \dots, h-1\} : \left\lfloor \frac{(r-1)d^{h-1}+s_t-1}{d^k} \right\rfloor = \left\lfloor \frac{(r-1)d^{h-1}+s_j-1}{d^k} \right\rfloor \right\} \end{aligned}$$

hold. This completes the proof of the formula claimed by this proposition.

Consider now two arbitrary leaves  $b_t, b_j$  of a  $d$ -regular tree and let  $u$  be their most recent common ancestor of level  $q$ . Then  $d_T(b_t, b_j) = 2(h-q)$  follows immediately from the definition of the most recent common ancestor. Together with  $d_T(b_t, b_j) = 2l$  with  $l$  is defined as in (3) we get  $q = \text{level}(u) = h-l$  and this completes the proof.  $\square$

**Definition 3** For a given arrangement  $\phi$  let  $B_u = \{\phi(1), \dots, \phi(n)\}$  be called the *set of used leaves*. If  $B_u = \{b_i, \dots, b_{i+n-1}\}$  holds for some  $1 \leq i \leq b-n+1$ ,  $\phi$  is called a *contiguous arrangement*.

Let us notice that not every instance of the DAPT possesses necessarily a contiguous optimal arrangement as illustrated by the following example.

*Example 1* A DAPT instance which does not possess any contiguous optimal arrangement.

The guest graph  $G$  with 12 vertices is represented in Fig. 3. Consider  $d = 4$ . In both pictures we identify the vertices with their indices, thus we write  $i$  instead of  $v_i$ ,  $i = 1, 2, \dots, 12$  for simplicity. Consider the non-contiguous (optimal) arrangement  $\phi$  represented in Fig. 4. The objective function value  $OV(G, 4, \phi)$  corresponding to  $\phi$  equals 28. For this instance the objective function value corresponding to an arbitrary arrangement  $\psi$  can be written as  $OV(G, 4, \psi) = 4 * a(\psi) + 2(m - a(\psi))$ , where  $m = 11$  is the number of edges of the guest graph and  $a(\psi)$  is the number of edges of  $G$  with end-vertices mapped by  $\psi$  into different basic subtrees of  $T$ . For the arrangement  $\phi$  we clearly have  $a(\phi) = 3$ .

We show now that for every contiguous arrangement  $\psi$ ,  $a(\psi) > 3$  holds, implying that  $OV(G, 4, \psi) > OV(G, 4, \phi)$ . Hence a contiguous arrangement can not be an optimal solution for this instance.

In order to see the above inequality we make a case distinction according to the number of neighbours of vertex  $v_1$  embedded together with  $v_1$  in the same basic subtree. Assume this number is 1 and w.l.o.g. vertex  $v_2$  is mapped together with  $v_1$  to the leaves of the same basic subtree, say  $T_1$ . Then of course  $v_4, v_7$  and  $v_{10}$  are not mapped by  $\psi$  into leaves of  $T_1$ . So  $a(\psi) \geq 3$ . Moreover, due to the contiguity of  $\psi$  for at least one of the paths  $\{v_4, v_5, v_6\}$ ,  $\{v_7, v_8, v_9\}$ ,  $\{v_{10}, v_{11}, v_{12}\}$  holds that not all of its vertices are mapped into the leaves of the same basic subtree. Due to that there is definitely one more edge (not incident to vertex  $v_1$ ) whose end-vertices are mapped by  $\psi$  into leaves of different basic subtrees, and hence  $a(\psi) \geq 4$ . The other cases where the number of neighbours

of  $v_1$  mapped together with  $v_1$  into the leaves of the same basic subtree is 2 or 3 can be argued upon analogously<sup>1</sup>.

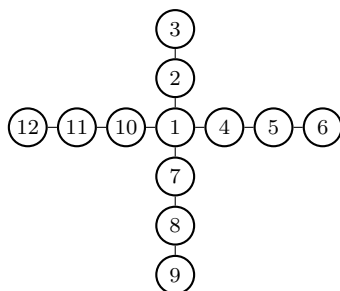


Fig. 3 The guest graph of the DAPT instance in Example 1.

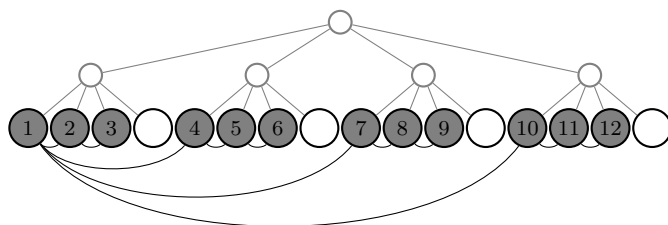


Fig. 4 The non-contiguous optimal arrangement which objective value is 28 for the DAPT instance in Example 1.

### 3 A lower bound

In a  $DAPT(G, d)$  with vertex set  $V(G)$  of size  $n$ ,  $n := |V(G)|$ , we have  $b := d^h$  leaves, where  $h = \lceil \log_d n \rceil$  is the height of the regular tree. Thus there are  $\frac{b!}{(b-n)!}$  possible arrangements and the complete enumeration becomes inefficient even for very small instances. Further let us notice that  $2m \leq OV(G, d, \phi) \leq 2hm$  holds for every arrangement  $\phi$ , where  $m$  is the number of edges of the guest graph  $G$ . These bounds are due to the fact that the distance between any two leaves in a regular tree of height  $h$  is between 2 and  $2h$ .

<sup>1</sup> In fact we can show that the DAPT is polynomially solvable in the case that the guest graph is an *extended star* and for some suitable choice of  $d$  as in this example. In this case the optimal arrangement has a particular structure and is in general not contiguous. This and other polynomially solvable special cases of the DAPT are discussed in another paper we are working in.

Next we introduce the so-called degree lower bound for the DAPT which will be also used to evaluate the performance of the heuristics introduced in this paper. We adapt an idea used by Petit (1998) for the linear arrangement problem. The idea is the construction of locally optimal arrangements for every vertex  $v$  of  $G$ , i.e. the construction of an optimal arrangement of  $v$  and its neighbours. Then the contribution of vertex  $v$  to the objective function value of any feasible solution cannot be smaller than the objective function value of this locally optimal arrangement divided by 2.

More precisely, for every  $v \in V(G)$  we define a new graph  $G'_v = (V'_v, E'_v)$  with the *vertex set*  $V'_v := V$  and the *edge set*  $E'_v = \{\{v, u\} : u \in \Gamma(v)\}$ . Thus  $G'_v$  is a subgraph of  $G$  containing all vertices of  $G$  and just the edges incident to  $v$ . Obviously,  $G'_v$  is the union of a *star* and some isolated vertices. An *optimal arrangement*  $\phi_v$  for  $DAPT(G'_v, d)$  is obtained by placing  $v$  on some *leaf*, say  $b_1$  w.l.o.g. and the other neighbours on the *leaves*  $b_2, \dots, b_{1+|\Gamma(v)|}$  one by one, where the canonical order of the leaves is adopted. The other vertices of  $G$  are arranged arbitrarily on the remaining leaves  $b_{2+|\Gamma(v)|}, \dots, b_{d^h}$ . Let  $OV_v$  denote the objective function value of the above mentioned arrangement for every  $v \in V$ . It is obvious that  $DB(G, d)$  given as below is a lower bound for  $DAPT(G, d)$ , that is

$$DB(G, d) = \frac{1}{2} \sum_{v \in V} OV_v \leq OV(G, d, \phi) \text{ for all arrangements } \phi. \quad (4)$$

This bound  $DG(G, d)$  is called the *degree bound*.

$DB(G, d)$  can be easily computed because  $OV_v$  can be easily computed, given  $d$  and the number  $|\Gamma(v)|$  of neighbours, for all  $v \in V(G)$ .

**Lemma 1** *Let  $G = (V, E)$  be a star graph with  $n$  vertices and  $2 \leq d \leq n$  a natural number. The optimal value  $OV$  of  $DAPT(G, d)$  is given as*

$$OV = 2 \left( h n - \frac{d^h - 1}{d - 1} \right), \quad (5)$$

where  $h = \lceil \log_d n \rceil$  is the height of the host  $d$ -regular tree.

*Proof* Let  $v := v_1$  be the central vertex of  $G$  with vertex set  $\{v_1, v_2, \dots, v_n\}$ . It is clear that the optimal arrangement places the vertices  $v_1, v_2, \dots, v_n$  into the leaves  $b_1, b_2, \dots, b_n$  of the  $d$ -regular tree of height  $h$ , respectively, where the leaves are given in the canonical order. Consider a partition of the set of leaves into sets  $B_j = \{b \text{ is a leaf} : d_T(b_1, b) = 2j\}$  with  $j = 0, \dots, h$ . It is clear that  $B_0 = \{b_1\}$ ,  $B_1 = \{b_2, \dots, b_d\}$ , and hence  $|B_0| = 1$ ,  $|B_1| = d - 1$ .

Generally, for  $j = 0, 1, \dots, h$ , a  $d$ -regular tree of height  $h$  contains  $d^{h-j}$   $d$ -regular subtrees of height  $j$ . Clearly one of these subtrees, say  $T_1$  contains  $b_1$ . This subtree has in turn  $d$   $d$ -regular subtrees of height  $j - 1$  and (only) one of those contains  $b_1$ . The set  $B_j$  consists exactly of the leaves of those  $d$ -regular subtrees of height  $j - 1$  of  $T_1$  which do not contain  $b_1$ . There are clearly  $d - 1$  such subtrees with  $d^{j-1}$  leaves each. Hence  $|B_j| = (d - 1)d^{j-1}$  for all  $j = 1, 2, \dots, h$ .



Because  $h = \lceil \log_d n \rceil$  we have  $d^{h-1} < n \leq d^h$  and hence the leaves of the basic subtree which contains  $b_1$  (and thus hosts  $v_1$ ) are all occupied. Consequently the other basic subtrees have exactly  $n - d^{h-1} > 0$  occupied leaves. Thus we get

$$OV = \sum_{j=1}^{h-1} 2j|B_j| + 2h(n - d^{h-1}) = 2(d-1) \sum_{j=1}^{h-1} jd^{j-1} + 2h(n - d^{h-1}).$$

Using  $\sum_{j=1}^{h-1} d^{j-1}j = \frac{((d-1)h-d)d^{h-1}+1}{(d-1)^2}$  we get the lemma.  $\square$

By applying Lemma 1 to evaluate  $OV_v$  in (4) as the optimal objective function value of the DAPT with the guest graph being the star graph with  $|\Gamma(v) + 1|$  vertices we get:

**Theorem 1** *Let  $G = (V, E)$  be a graph and  $2 \leq d \leq n$  the degree of the host tree. Then the degree bound is given as*

$$DB(G, d) = \sum_{v \in V} \left( p(v)(|\Gamma(v)| + 1) - \frac{d^{p(v)} - 1}{d - 1} \right) \quad (6)$$

where

$$p(v) := \lceil \log_d (|\Gamma(v)| + 1) \rceil. \quad (7)$$

## 4 Heuristic approaches for the DAPT

In this section we will introduce some simple greedy heuristics, a construction heuristic and two local search heuristics for the DAPT.

### 4.1 Simple greedy approaches

A simple greedy strategy considers the leaves of the host graph in the canonical order. The first leaf is occupied by a vertex selected at random. Then we consider the next leaf in the canonical order, place there the “best possible vertex”, and repeat this process until all vertices of the guest graph have been placed to some leaf. “The best possible vertex” means here a vertex which leads to the *biggest increase* in the objective function value of the DAPT. We call this heuristic G2. G2 is a leaf-driven heuristic. Notice that this heuristic yields always a contiguous arrangement. Clearly there are also vertex-driven greedy algorithms which investigate the vertices of the guest graph in some prespecified order and place the current vertex to the “best possible free leaf”. Since the vertex-driven greedy heuristics we have tested were outperformed by the leaf-driven greedy heuristic described above we do not present them in details in this paper.

The time complexity of G2 is  $O(\max\{(m+n)n, n^2 \log n\})$ . To see this consider first a pre-processing step to compute the distances between all pairs of leaves of the arrangement tree in  $O(n^2 \log n)$  time according to Observation 1. Then  $n$  iterations are performed to arrange the vertices one at a time. The computation of the increase in the objective function value resulting by placing a specific vertex  $v$  onto the current leaf takes  $O(|\Gamma(v)|)$  time per each vertex and hence  $O(m)$  time for all candidate vertices. Selecting the best among all candidate vertices takes another  $O(n)$  time. Thus we obtain a time complexity of  $O(n+m)$  per iteration which results to  $O((n+m)n)$  for all iterations and to an overall time complexity of  $O(\max\{(m+n)n, n^2 \log n\})$  (including the pre-processing step).

We have also tested two very simple search heuristics BFSG and DFSG which order the vertices of the guest graph according to breadth-first search or depth-first search, respectively, after starting at some prespecified vertex. Then the vertices are placed onto the leaves in the canonical order, i.e. the  $i$ -th vertex according to the resulting ordering is placed at the  $i$ -th leaf,  $i = 1, 2, \dots, n$ .

Of course there are a number of variants of this algorithm. We distinguish different implementations for connected and non-connected graphs. In the case of a connected guest graph  $G$  there is a flexibility in choosing the starting vertex for search algorithm in  $G$ . Depending on the graph structure the vertex with the highest degree can be chosen. Or the algorithm is run for each vertex as starting vertex and then the best solution obtained is chosen.

In the case of non-connected graphs we have to fix the order of the connected components before running the search algorithm for each of them. This can be done in many ways, e.g. by considering the connected components in decreasing order of magnitude.

Clearly, the worst-case time complexity depends on the particular implementation in each case. In the case of connected graphs we obtain an  $O(n^3)$  algorithm, if the “best” starting vertex among all is chosen. In the case of non-connected graphs we obtain the same time complexity, if we choose the best starting vertex in each component by running the algorithm as many times as the number of vertices for each component.

## 4.2 A construction heuristic

Let us now consider the objective function of the problem from another point of view. Let  $a_i$ ,  $1 \leq i \leq h$ , be the number of edges of the guest graph  $G$  whose endpoints are mapped into leaves of  $T$  at a distance  $2i$  in the host graph.

We can state the obvious fact that

$$OV(G, d, \phi) = 2ha_h + 2(h-1)a_{h-1} + \dots + 2a_1, \quad (8)$$

where  $a_h + a_{h-1} + \dots + a_1 = m$  and  $m$  is the number of edges of the guest graph  $G$ .

Since our aim is to minimise the objective value  $OV(G, d, \phi)$ , we try first to minimise the coefficient  $a_h$  by partitioning the vertex set  $V$  in at most  $d$

subsets  $V_i$ ,  $1 \leq i \leq d$ , with  $0 \leq |V_i| \leq \frac{|B|}{d}$ . Then each  $V_i$ ,  $1 \leq i \leq d$ , is embedded into the leaves of the corresponding basic subtree, which means that the inequalities  $(i-1)d+1 \leq \phi(v) \leq id$  hold for any  $v \in V_i$ ,  $1 \leq i \leq d$ . Among all arrangements of this kind we choose one which minimises  $a_h = |\{(u,v) \in E | u \in V_i, v \in V_j, i \neq j\}|$ . Then the subproblems  $DAPT(G[V_i], d)$ ,  $1 \leq i \leq d$ , (where  $G[V_i]$  is the subgraph of  $G$  induced by the set of vertices  $V_i$ ) are solved in order to determine an arrangement of  $V_i$ ,  $1 \leq i \leq d$ , into the leaves of the corresponding basic subtree.

The problem of partitioning  $V$  as described above is strongly related to the so called *minimum cut problem with bounded set size (MCBSSP)* described in the next subsection. In Subsection 4.2.2 we present an approach to solve the  $DAPT(G, d)$  by using the idea described above and a heuristic for MCBSSP.

#### 4.2.1 A related problem (MCBSSP) and some heuristic approaches

##### **The Minimum Cut Problem with Bounded Set Size (MCBSSP)**

Input: A graph  $G = (V, E)$  with  $n = |V|$  and two integers  $l, u$  with  $0 < l \leq u < n$ .

Output: A set  $X \subset V$  with  $l \leq |X| \leq u$  such that the cut

$\delta(X) := \{(u, v) \in E | u \in X, v \notin X\}$  has minimum cardinality.

MCBSSP is equivalent to the so-called  $(k, n-k)$  cut problem ( $k$ - $(n-k)$ CP), investigated by Feige, Krauthgamer and Nissim in Feige et al (2003).

##### **The $(k, n-k)$ cut problem ( $k$ - $(n-k)$ CP)**

Input: A graph  $G = (V, E)$  with  $n = |V|$  and an integer  $k$ , with  $k < n$ .

Output: A partition of  $V$  in  $X, Y$  with  $|X| = k$ ,  $|Y| = n - k$  such that the cut

$\delta(X) := \{(u, v) \in E | u \in X, v \in Y\}$  has minimum cardinality.

Indeed the equivalence between MCBSSP and  $k$ - $(n-k)$ CP is trivial: an optimal solution of MCBSSP in a graph  $G$  with input parameters  $l, u$  can be obtained by solving  $O(n)$  instances of  $k$ - $(n-k)$ CP in the same graph  $G$  with input parameter  $k = l, l+1, \dots, u$ . On the other hand  $k$ - $(n-k)$ CP is just a special case of MCBSSP, when  $u = l$  holds.  $k$ - $(n-k)$ CP is NP-hard for general  $k$  as mentioned in Feige et al (2003), a special case of it is the *minimum bisection problem*, see Garey and Johnson (1979). Thus MCBSSP is also NP-hard for general  $l$  and  $u$  and there is no hope to optimally solve it in polynomial time (unless  $P = NP$ ).

We have considered two heuristic approaches to solve MCBSSP. These will then be applied recursively to obtain a heuristic for the  $DAPT(G, d)$  as described above.

The first approach is based on a polynomial time approximation algorithm for  $k$ - $(n-k)$ CP with an approximation ratio  $O(\log^2 n)$  proposed by Feige et al (2003). (Their algorithm reaches an even better approximation rate for the case  $k = O(\log n)$ ). So in order to obtain a solution of MCBSSP in the graph

$G$  with parameters  $l$  and  $u$  we apply the approach of Feige et al (2003) to  $k$ -( $n - k$ )CP in  $G$  with parameter  $k$  varying between  $l$  and  $u$  and then choose a minimum cut among the  $l - u + 1$  obtained solutions of  $k$ -( $n - k$ )CP. Since  $u - l \leq n$  we get a polynomial time approach for MCBSSP.

Our second approach for MCBSSP makes use of a simple *local search idea*. Assume that  $l = u$ . We randomly partition  $V$  in  $X$  and  $V \setminus X$ , where  $\emptyset \subset X \subset V$  and  $|X| = l = u$ . We try to decrease the cardinality of the cut  $|\delta(X)|$  by the following pair-exchange approach. Consider another cut  $\delta((X \setminus \{u\}) \cup \{v\})$  for each pair  $(u, v)$ , where  $u \in X$  and  $v \notin X$ . Replace  $X$  by  $(X \setminus \{u\}) \cup \{v\}$  if  $\delta((X \setminus \{u\}) \cup \{v\}) < \delta(X)$  and repeat this step until no further improvement of the cardinality of the cut is possible. Then apply the above approach to determine a cut  $\delta(X^{(k)})$  with  $|X^{(k)}| = k$  for any  $l \leq k \leq u$  and choose the best among the cuts  $\delta(X^{(k)})$ ,  $l \leq k \leq u$ .

#### 4.2.2 A heuristic for $DAPT(G, d)$

Having described the heuristics for MCBSSP let us turn back to the  $DAPT(G, d)$ . The approach is presented in the form of a pseudo code in Algorithm 4.1 and involves the heuristic solution of the MCBSSP as a subroutine (see pseudocode line 11).

We first consider the question of determining the “unused leaves”, i.e. leaves of the arrangement tree, into which no vertices of the guest graph are arranged. Based on our observations in the context of numerical tests we try to use as few basic subtrees as possible to arrange all vertices of the guest graph. Thus we collect the unused  $b - n$  leaves (recall that  $b := |B|$  is the number of leaves of the host  $d$ -regular tree) into as few basic subtrees as possible. By considering that each basic subtree has  $b_1 := \frac{b}{d}$  leaves, we mark the first  $l_{uu} = \left\lfloor \frac{b-n}{b_1} \right\rfloor b_1$  leaves, or equivalently the first  $\left\lfloor \frac{b-n}{b_1} \right\rfloor$  basic subtrees as *unused* (see pseudocode lines 7 – 9). Then we separate the vertices  $X$  which will be placed on the leaves  $b_{l_{uu}+1}, \dots, b_{l_{uu}+\frac{b}{d}}$ , i.e. on the leaves of the first used basic subtree, by solving MCBSSP with the parameters  $l := b_1 - (b - n) \bmod b_1$  and  $u := b_1$  (see pseudocode line 11). This can be done by applying one of the heuristics described in Subsection 4.2.1. We repeat then this procedure  $\left\lceil \frac{n}{b_1} \right\rceil - 1$  times to obtain  $\left\lceil \frac{n}{b_1} \right\rceil$  subproblems which are solved recursively (pseudocode line 12). The recursion calls will terminate when the height of the *arrangement tree* becomes 1; there an *arrangement*  $\phi$  is selected at random.

Now let us consider the worst-case time complexity of the described approach. Let  $f_C(n)$  denote the worst-case time complexity of the subroutine which solves MCBSSP for a graph with  $n$  vertices and any parameters  $0 < l \leq u < n$ . Since  $n \leq b$  holds for all instances, the *worst-case time*

**Input:**  $G = (V, E)$  undirected graph and positive integer  $d \in \mathbb{N}$  where  $2 \leq d \leq n$ ; let be  $|V| = n$  and  $T$  the  $d$ -regular arrangement tree with the set of leaves  $B$

**Output:** arrangement  $\phi : V \rightarrow B$

- 1:  $h := \lceil \log_d n \rceil$  and  $b := h^d$ ;
- 2: **if**  $h = 1$  **then**
- 3:   make the arrangement  $\phi$  at random;
- 4: **else**
- 5:    $l_{uu} := b - n$ ;
- 6:   **for**  $i := 1$  **to**  $d$  **do**
- 7:     **if**  $l_{uu} \geq \frac{b}{d}$  **then**
- 8:       $\phi^{-1}(l) := \text{unused}, (i-1)\frac{b}{d} \leq l \leq i\frac{b}{d}$ ;
- 9:       $l_{uu} := l_{uu} - \frac{b}{d}$ ;
- 10:    **else**
- 11:     find a minimum cardinality cut  $X \subset V(G)$  in graph  $G$  subject to  $\frac{b}{d} - l_{uu} \leq |X| \leq \frac{b}{d}$  by solving MCBSSP with parameters  $l := \frac{b}{d} - l_{uu}$  and  $u := \frac{b}{d}$ ;
- 12:     solve the problem for the graph  $G[X]$  and a  $d$ -regular arrangement tree  $T_X$  which height is  $h-1$  recursively; let  $\phi_X$  be the solution of this recursive problem;
- 13:     compute the inverse function of  $\phi_X$  which we denote  $\phi_X^{-1}$ ;
- 14:     **for**  $j := 1$  **to**  $\frac{b}{d}$  **do**
- 15:       $\phi^{-1}((i-1)\frac{b}{d} + j) := \phi_X^{-1}(j)$ ;
- 16:     **end for**
- 17:      $G := G[G \setminus X]$ ;
- 18:      $l_{uu} := l_{uu} - (\frac{b}{d} - |X|)$ ;
- 19:    **end if**
- 20:   **end for**
- 21:   compute the function  $\phi$  from the function  $\phi^{-1}$ ;
- 22: **end if**

Algorithm 4.1: Construction heuristic.

*complexity* of the whole algorithm is

$$\begin{aligned}
& 1 \left( f_C \left( \frac{b}{d}d \right) + f_C \left( \frac{b}{d}(d-1) \right) + \dots + f_C \left( \frac{b}{d}2 \right) \right) + \\
& d \left( f_C \left( \frac{b}{d}d \right) + f_C \left( \frac{b}{d}(d-1) \right) + \dots + f_C \left( \frac{b}{d}2 \right) \right) + \\
& \dots \\
& d^{h-2} \left( f_C \left( \frac{b}{d^{h-2}}d \right) + f_C \left( \frac{b}{d^{h-2}}(d-1) \right) + \dots + f_C \left( \frac{b}{d^{h-2}}2 \right) \right),
\end{aligned} \tag{9}$$

where the lines correspond to the *recursion depth*. Notice that if the height of the *arrangement tree* is 1, the arrangement  $\phi$  can be made at random and thus the *recursion depth* is only  $h-2$ . Summarising we get the following worst case time complexity

$$\sum_{i=0}^{h-2} d^i \sum_{j=0}^{d-2} f_C \left( \frac{b}{d^{i+1}}(d-j) \right). \tag{10}$$

For some particular heuristic to solve the MCBSSP we can substitute  $f_C(n)$  by a precise expression in (10). Consider the case of the *local search* based

heuristic described in Subsection 4.2.1. When computing the cuts at the first recursion level  $u - l \leq \frac{b}{d}$  obviously holds. If  $X$  is the set of vertices generating the cut, then  $|X| \leq \frac{b}{d}$  holds. When computing the  $k$ -th cut at the first recursion level we have at most  $\frac{b}{d} \frac{(d-k)b}{d}$  vertex pairs which could be exchanged and the cardinality of the cut after the pair-exchange can be computed in  $O(\frac{b}{d} + (d-k)\frac{b}{d}) = O(\frac{b}{d}(d-k+1))$  time. So we get a worst-case time complexity of  $O\left(\frac{b}{d} \left(\frac{b}{d} \frac{(d-k)b}{d}\right) \left(\frac{b}{d} + \frac{(d-k)b}{d}\right)\right) = O\left(\frac{b}{d} \left(\frac{b}{d} \frac{(d-k)b}{d}\right) \frac{b}{d}(d-k+1)\right) = O\left(\frac{b}{d}\right)^4 (d-k)(d-k+1)$  for the  $k$ -th cut in the first level (where the first factor in the above expression accounts for the number of  $k$ - $(n-k)$ CP to be solved which is at most  $u - l \leq \frac{b}{d}$ ). Summarising for all cuts of the first level we get

$$\begin{aligned} O\left(\left(\frac{b}{d}\right)^4 \sum_{k=1}^{d-1} ((d-k)(d-k+1))\right) &= O\left(\left(\frac{b}{d}\right)^4 \left(\sum_{i=1}^{d-1} i^2 + \sum_{i=1}^{d-1} i\right)\right) \\ &= O\left(\frac{b^4}{d}\right). \end{aligned} \quad (11)$$

Now let us consider the recursion. After building the first  $d - 1$  cuts we get  $d$  subproblems each of them having most  $\frac{b}{d}$  vertices. Thus for the whole algorithm we get a time complexity  $K$  with

$$K := O\left(\frac{b^4}{d} + d \frac{\left(\frac{b}{d}\right)^4}{d} + d^2 \frac{\left(\frac{b}{d^2}\right)^4}{d} + \dots + d^{h-2} \frac{\left(\frac{b}{d^{h-2}}\right)^4}{d} + n\right). \quad (12)$$

Using  $d^h \frac{\left(\frac{b}{d^h}\right)^4}{d} = b \frac{\left(\frac{b}{d}\right)^4}{d} = \frac{b}{d}$ ,  $d^{h-1} \frac{\left(\frac{b}{d^{h-1}}\right)^4}{d} = \frac{b}{d} \frac{\left(\frac{b}{d}\right)^4}{d} = bd^2$  and considering  $b < nd$  we get

$$K = O\left(\frac{b^4}{d} \sum_{i=0}^h \left(\frac{1}{d^3}\right)^i - bd^2 - \frac{b}{d} + n\right) = O(n^4 d^3). \quad (13)$$

Now, we can state the following theorem.

**Theorem 2** *The Algorithm 4.1 can be implemented with a worst case time complexity of  $O(n^4 d^3)$ , if the local search approach of Subsection 4.2.1 is applied to solve MCBSSP.*

In fact the quality of this construction heuristic depends significantly on the quality of the heuristic used to solve MCBSSP. However, even if we were able to solve MCBSSP to optimality, the construction heuristic would not necessarily compute an optimal arrangement. As an example consider  $DAPT(G, 2)$  with guest graph  $G$  as shown in Fig. 5. Fig. 6 shows an arrangement obtained by the construction heuristic, where MCBSSP was always solved to optimality during the algorithm. This arrangement is not optimal; a strictly better arrangement is shown in Fig. 7 (this is actually an optimal arrangement). The reason for this

behaviour relies on the fact that minimising the coefficients  $a_i, i = 1, 2, \dots, h$ , starting with  $a_h$  and proceeding in the above order, does not necessarily lead to a minimum value of  $OV(G, d, \phi)$ , see (8).

In our computational experiment we observed that the construction heuristic which involved the pair-exchange approach to solve MCBSSP outperforms the heuristic which involves the approach of Feige et al (2003). Therefore in Section 6 we just report on the performance of the more successful algorithm denoted by CHLS, see also Section 6.4.

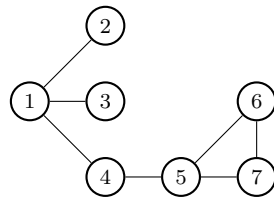


Fig. 5 Graph.

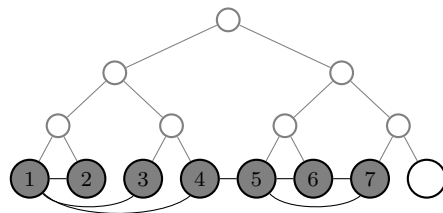


Fig. 6 A non-optimal arrangement  $\phi$  with  $OV(G, 2, \phi) = 26$  for  $G$  in Fig. 5.

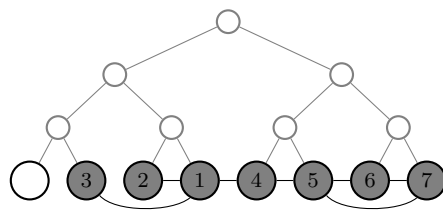


Fig. 7 An optimal arrangement  $\phi$  with  $OV(G, 2, \phi) = 24$  for  $G$  in Fig. 5.

### 4.3 Local search approaches

In this paragraph we propose two different local search heuristics for the DAPT. They can be used separately or also combined as described below.

#### 4.3.1 The pair-exchange heuristic

The algorithm starts with an arbitrary arrangement  $\phi$  (it can be a random arrangement or an arrangement obtained by applying some other heuristic) and tries to improve the objective function value by performing so-called pair-exchanges. More precisely the algorithm fixes an ordering in which the pairs of vertices  $(v_i, v_j) \in V(G) \times V(G)$  with  $v_i \neq v_j$  are considered for a pair exchange. The algorithm checks whether a pair  $(v_i, v_j)$  exists such that  $OV(G, d, \phi') < OV(G, d, \phi)$ , where  $\phi'$  is obtained from  $\phi$  by applying a pair-exchange:

$$\phi'(v_k) = \begin{cases} \phi(v_j) & \text{if } k = i \\ \phi(v_i) & \text{if } k = j \\ \phi(v_k) & \text{if } k \notin \{i, j\} \end{cases} \quad \text{for } k \in \{1, \dots, n\}. \quad (14)$$

If such a pair  $(v_i, v_j)$  of vertices whose exchange improves the objective function value can be found, then  $\phi$  is substituted by  $\phi'$  and the procedure is iteratively repeated. Otherwise the algorithm terminates and outputs the current arrangement. Notice that this approach keeps unchanged the set of unused leaves.

**Theorem 3** *The pair-exchange heuristic for the DAPT( $G, d$ ) can be implemented with time complexity  $O(n^2 m \min\{m, n\}(\log n))$ , where  $n$  is the number of vertices and  $m$  is the number of edges in  $G$ .*

*Proof* There are  $O(b^2)$  pairs of vertices in the graph  $G'$ . Since  $2m \leq OV(G, d, \phi) \leq 2hm$  holds for every arrangement  $\phi$ , we can make at most  $O(2hm - 2m) = O(hm) = O((\log b)m) = O(\log(nd)m) = O((\log n + \log d)m) = O(m \log n)$  improvements of the objective function value (if  $d$  is considered to be a constant and by using  $b < nd$ ).

Consider that the pairwise distances between all pairs of leaves in the arrangement tree can be computed in  $O(b^2 \log n) = O(n^2 d^2 \log n)$  time in a pre-processing step, see Observation 1. In order to update the objective function value of an arrangement after a pair-exchange of vertices  $v_i$  and  $v_j$  which transforms the current arrangement  $\phi$  to the arrangement  $\phi'$  as in (14), the length of the path between  $\phi(v_i)$  ( $\phi(v_j)$ ) and  $\phi(v)$  is substituted by the length of the corresponding path between  $\phi'(v_i)$  ( $\phi'(v_j)$ ) and  $\phi'(v)$ , for all neighbours  $v$  of  $v_i$  ( $v_j$ ). Since the vertices which exchange position have in total  $O(\min\{m, n\})$  neighbours, the objective function after a (candidate) pair-exchange can be updated in  $O(\min\{m, n\})$  time. With at most  $O(b^2)$  (candidate) pair-exchanges to be performed in each iteration and at most  $O(m \log n)$  iterations, the overall time complexity of the algorithm amounts to  $O(b^2 \min\{m, n\} m \log n) = O(n^2 d^2 m \min\{m, n\} \log n)$ .  $\square$



Clearly, we can also fix an ordering of the pairs of leaves and exchange the vertices arranged at some pair of leaves (if any), in this ordering. One would obtain a similar time complexity as in the general case of Theorem 3. We refer to these heuristics as *vertex-based pair-exchange heuristic* and *leaf-based pair-exchange heuristic*, respectively. Our computational experiments have shown that the vertex-based pair-exchange heuristic generally outperforms the leaf-based pair-exchange heuristic. For this reason we only report about the performance of the vertex-based pair-exchange heuristic (abbreviated by PEHVNA) in Section 6.

#### 4.3.2 The shift-flip heuristic

The last heuristic we discuss is the *shift-flip heuristic*. First, we need two definitions.

**Definition 4 (Flip)** Let  $G = (V, E)$  be a guest graph with  $|V| = n$ ,  $T$  a  $d$ -regular tree, with  $2 \leq d \leq n$ , and let  $B$  be the set of leaves of  $T$ . Let  $\phi : V \rightarrow B$  be an arrangement. Further, let  $e, g, l, r \in \mathbb{N} \cup \{0\}$ , be parameters with  $0 \leq e < h$ ,  $1 \leq g \leq d^e$ ,  $1 \leq l < r \leq d$ . Finally let  $f$  be a bijection  $f : B \rightarrow B$  defined as follows:

$$f(b_i) = \begin{cases} b_{\Delta(g)+(r-1)d^{h-(e+1)+t_i}} & \text{for } i = \Delta(g) + (l-1)d^{h-(e+1)} + t_i, 1 \leq t_i \leq d^{h-(e+1)} \\ b_{\Delta(g)+(l-1)d^{h-(e+1)+t_i}} & \text{for } i = \Delta(g) + (r-1)d^{h-(e+1)} + t_i, 1 \leq t_i \leq d^{h-(e+1)} \\ b_i & \text{otherwise} \end{cases}, \quad (15)$$

where  $\Delta(g) := (g-1)d^{h-e}$ . The arrangement  $\phi_f : V \rightarrow B$  where  $\phi_f = f \circ \phi$  is a *flip of the arrangement*  $\phi$ . We say that we *flip* the arrangement  $\phi$  at the  $l$ -th and  $r$ -th  $d$ -regular subtrees of the  $g$ -th vertex in level  $e$ .

In a more descriptive explanation a flip consists of interchanging the preimages of the leaves of two  $d$ -regular subtrees of the arrangement tree which have the same height and whose roots have a common father vertex, while preserving the order of the leaves in each of the two interchanged subtrees. More precisely we consider the vertices of the  $d$ -regular tree as being partitioned into levels, the root having level 0, its  $d$  children having level 1 and so on, to end up with the leaves at level  $h$ . In Definition 4 we consider the  $g$ -th vertex in level  $e$  and the indices  $l$  and  $r$  of two children of that vertex. The successors of each of those children build a  $d$ -regular subtree of height  $h - (e + 1)$ , respectively. The flip operation interchanges exactly the preimages of the leaves of these two  $d$ -regular subtrees by preserving in each subtree the order of the leaves induced by the canonical order of the leaves in  $T$ .

For an illustration consider an instance  $DAPT(G, d)$  with guest graph  $G$  given in Fig. 8 and  $d = 3$ .

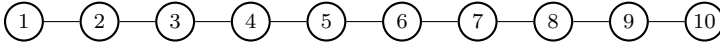


Fig. 8 A guest graph.

Consider further an arrangement represented in Fig. 9; each filled leaf contains the index of the vertex of  $G$  mapped into that leaf.

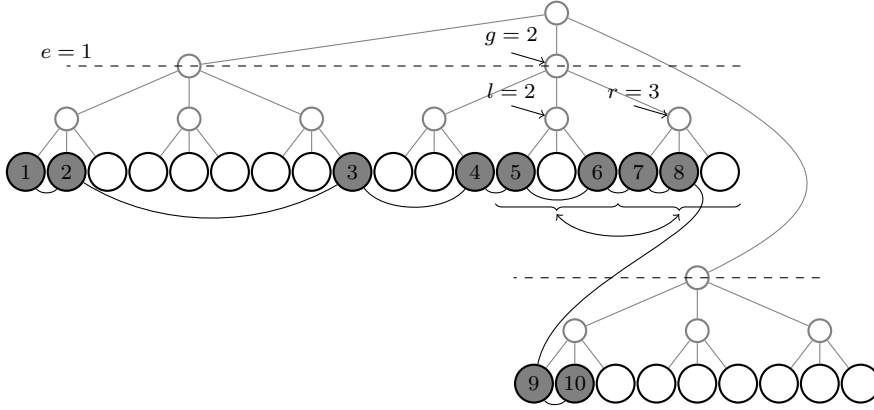


Fig. 9 An arrangement  $\phi$  with  $OV(G, 3, \phi) = 32$  for  $G$  in Fig. 8.

In Fig. 10 we see the flip obtained from the arrangement represented in Fig. 9 with parameters  $e = 1$ ,  $g = 2$ ,  $l = 2$  and  $r = 3$ . Notice that flipping does not change the objective function value of the arrangement.

**Proposition 1** Let  $G = (V, E)$  be an undirected guest graph with  $|V| = n$ ,  $T$  a  $d$ -regular tree, with  $2 \leq d \leq n$  and let  $B$  be the set of leaves of  $T$ . Further, let  $e, g, l, r \in \mathbb{N} \cup \{0\}$ , be parameters with  $0 \leq e < h$ ,  $1 \leq g \leq d^e$ ,  $1 \leq l < r \leq d$ . Let  $f$  be a bijective function  $f : B \rightarrow B$  defined as in Definition 4. For any arrangement  $\phi : V \rightarrow B$  and the corresponding flip  $\phi_f : V \rightarrow B$  of the arrangement  $\phi$ ,  $\phi_f = f \circ \phi$ , the equality  $OV(G, d, \phi_f) = OV(G, d, \phi)$  holds.

*Sketch of the proof*

Since

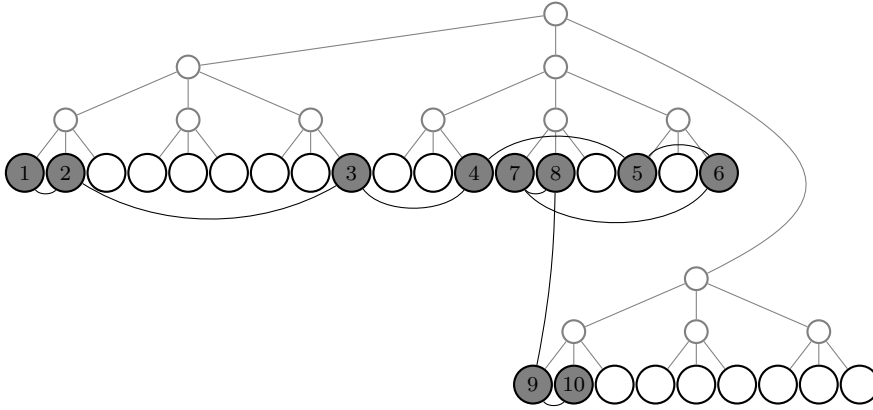
$$OV(G, d, \phi) = \sum_{\{v_i, v_j\} \in E(G)} d_T(\phi(v_i), \phi(v_j))$$

and

$$OV(G, d, \phi_f) = \sum_{\{v_i, v_j\} \in E(G)} d_T(\phi_f(v_i), \phi_f(v_j)),$$

the claim would follow immediately from the equalities

$$d_T(\phi(v_i), \phi(v_j)) = d_T(\phi_f(v_i), \phi_f(v_j)), \text{ for all edges } (v_i, v_j) \text{ of } G. \quad (16)$$



**Fig. 10** A flip of the arrangement shown in Fig. 9, for the guest graph shown in Fig. 8 and  $d = 3$ . The parameters of the flip are  $e = 1$ ,  $g = 2$ ,  $l = 2$  and  $r = 3$ . The objective value of the flipped arrangement remains unchanged and equals 32.

According to Observation 1 the property (P) below would imply the later inequalities:

(P) For every edge  $(v_i, v_j)$  of the guest graph  $G$  the most recent ancestor of  $\phi(v_i)$ ,  $\phi(v_j)$  coincides with the most recent ancestor of  $\phi_f(v_i)$ ,  $\phi_f(v_j)$ .

It might be intuitively clear that property P holds. To see that the following two (relevant) cases could be considered separately: (I)  $\phi(v_i) \neq \phi_f(v_i)$  and  $\phi(v_j) \neq \phi_f(v_j)$ , and (II) either  $\phi(v_i) \neq \phi_f(v_i)$  or  $\phi(v_j) \neq \phi_f(v_j)$ .

A rigorous and detailed proof of equalities (16) is provided in the appendix.

**Definition 5 (shift)** Let  $G = (V, E)$  be an undirected guest graph with  $|V| = n$  and  $T$  a  $d$ -regular arrangement tree with  $2 \leq d \leq n$ , set of leaves  $B$  and number of leaves  $b = |B|$ . Let  $\phi : V \rightarrow B$  be an arrangement. Further, let  $k \in \mathbb{N}$  be an integer. An arrangement  $\phi_k$  with

$$\phi_k(v) := b_{((i-1)+k) \bmod b + 1}, \text{ where } \phi(v) = b_i, \quad (17)$$

is a *shift* of the arrangement  $\phi$ . We say that we *shift* the arrangement  $\phi$  by  $k$ .

The idea of the shift-flip heuristic is fairly simple. For a given arrangement  $\phi$  we find out a  $1 \leq k \leq b$  which minimises the objective function value  $OV(G, d, \phi_k)$ . There are two possibilities to define the shift step. In the first variant we apply the shift by  $k$  defined as above only if it implies an improvement of the objective function value, i.e.  $OV(G, d, \phi_k) < OV(G, d, \phi)$ , and substitute then the current arrangement  $\phi$  by the improved one  $\phi_k$ . In the second variant we also accept an arrangement which keeps the objective function value unchanged. If no such an arrangement can be found, then a further flip is performed. Both approaches proceed in the next iteration by applying a

random flip to the current arrangement and so on until a termination criterion is satisfied. Both variants of the heuristic output the best arrangement found during the search. We report about the performance of the second variant because this variant seems to outperform the first one.

Of course there are a number of possibilities to define a terminating criterion. It can be a run time bound which defines the maximum length of a time interval the algorithm is allowed to run without doing an improvement. Or it can be a bound on the overall number of flip and shift steps performed without improving the objective function value.

Both variants of the shift-flip heuristic (SF) can be combined with the pair-exchange heuristic (PE). Since the search neighbourhoods of the two heuristics are significantly different, it is possible to escape from the local minima of SF by just applying a search in the PE neighbourhood and vice-versa.

## 5 Test instances

We test and compare the above described heuristics on some families of test instances. To the best of our knowledge there are no standard test instances for this problem, so we have generated some test instances ourselves. We introduce the following families of test instances which are also available at <http://www.opt.math.tu-graz.ac.at/~cela/public.htm> and at <http://www.rostislavstanek.at/daten/DAPTLIB.zip>.

### 5.1 Test instances solvable by complete enumeration

The guest graphs of these instances are marked by the prefix “*CE*”. The first graph in this category *CE\_sample* corresponds to the graph in Fig. 1. Further we consider 2 (sparse) graphs *CE\_sparse7* ( $n = 7, m = 7$ ) and *CE\_sparse10* ( $n = 10, m = 11$ ) with 7 and 10 vertices, respectively. We generate test instances with guest graph *CE\_sparse7* and all possible values of  $d, 2 \leq d \leq 7$ . With the guest graph *CE\_sparse10* we generate instances with  $d = 2$  and  $d = 4$ . Further we consider some analogous instances with denser guest graphs: *CE\_dense7* ( $n = 7, m = 14$ ) and *CE\_dense10* ( $n = 10, m = 26$ ) with 7 and 10 vertices. Finally, we consider a  $3 \times 3$  mesh *CE\_mesh9* and  $d = 2, d = 3$  and  $d = 4$ .

For this family of test instances the precise values of  $d$  were chosen so as to be able to solve these instances by complete enumeration within a prespecified time limit, see Section 6.

### 5.2 Test instances with known optimal solution

These instances are special cases of the DAPT which can be solved by a polynomial time algorithm, see ela and Staněk (2013) and Staněk (2012). The guest graphs of these instances are marked by the prefix “*SC*”. Unless

the special case involves a particular choice of  $d$ , we use  $d = 2$  and  $d = 7$  for all considered guest graphs. We consider instances of following types:

- Instances for which  $d = n - 1$  where  $n$  is the number of vertices of the guest graph. We use 3 guest graphs generated at random for this class of instances: *SC\_random25*, *SC\_random50* and *SC\_random75*. These graphs have the same number of vertices,  $n = 500$ , and in each of them any pair of non-equal vertices build an edge independently at random with probability 0.25, 0.50 and 0.75, respectively.
- Instances whose guest graphs build a *star*, that is they consist just of a *central vertex* connected by an edge to all other vertices of the graph. The concrete graphs are *SC\_star50*, *SC\_star500* and *SC\_star1000* with 50, 500 and 1000 vertices, respectively.
- The guest graph in Fig. 3 and the choice  $d = 4$ . This guest graph is an extended star and this instance is referred to as *SC\_extStar*.
- Instances whose guest graphs build a *d-regular tree*. We denote these guest graphs/instances by *SC\_treeDGxHy* where  $x = d$  holds and  $y$  is the height of the tree.
- Instances whose guest graphs build of a *path*. We denote these guest graphs by *SC\_path50*, *SC\_path500* and *SC\_path1000*. They have 50, 500 and 1000 vertices, respectively.
- Instances whose guest graphs build a *simple cycle*. We created 3 graphs of this type: *SC\_simpleCycle50*, *SC\_simpleCycle500* and *SC\_simpleCycle1000* with 50, 500 and 1000 vertices, respectively.

### 5.3 Randomly generated test instances

The guest graphs of these instances are marked by the prefix “*RG\_*”. This instances are generated in the same way as the instances *SC\_random25*, *SC\_random50* and *SC\_random75*. All guest graphs in this class of instances have 500 vertices and the pairs of vertices are present as edges in the graphs randomly and independently with the same constant probability, say  $\frac{x}{100}$ . For each  $x$  two random graphs are constructed as above and are denoted by *RG\_randomAx* and *RG\_randomBx*. The degree of the regularity of the host tree is set to  $d = 2$  and  $d = 7$ .

### 5.4 Instances with graphs taken from Petit Petit (2003)

The guest graphs of these instances are marked by the prefix “*Pet03\_*”. These graphs were used in Petit (2003) to test some heuristics for the linear arrangement problem (LAP), a problem related to the DAPT as explained in Section 1. Also in this family of instances we use  $d = 2$  and  $d = 7$ . This choice of the parameter  $d$  is motivated by the goal of comparing the behaviour of the proposed heuristics when a smaller and a larger value of the parameter  $d$  are considered ( $d = 2$  and  $d = 7$ ).

## 6 Numerical results

The results of all numerical tests are summarised in the tables in the appendix. We group the test instances described in Section 5 in three groups: instances solvable by complete enumeration, polynomially solvable instances and the rest. Table 2 reports on instances which could be solved to optimality by complete enumeration on the following computer in 1 week: HP Compaq nx7400, 32 bit Intel processor (Intel®Centrino® Duo T2250 1.73 GHz), running in Ubuntu (Linux). Table 3 summarises the results for the instances which are solvable to optimality in polynomial time. Table 4 summarises the computational results obtained for the remaining instances.

In order to compare the quality of the proposed heuristics we define a *quality quotient* as follows

$$q(\mathcal{I}, \mathcal{H}) = \frac{1}{|\mathcal{I}|} \sum_{DAPT(G,d) \in \mathcal{I}} \frac{\min_{HE \in \mathcal{H}} \{HE(G, d)\}}{\max \{OS(G, d), DG(G, d)\}}, \quad (18)$$

where  $\mathcal{I}$  denotes a set of test instances  $DAPT(G, d)$  with guest graph  $G$  and degree of regularity  $d$ .  $\mathcal{H}$  denotes a set of heuristics,  $HE(G, d)$  stands for the objective value obtained from the heuristic  $HE$  for the instance  $DAPT(G, d)$ ,  $DG(G, d)$  represents the degree bound for this instance and  $OS(G, d)$  stands for the objective function value of an optimal solution. We set  $OS(G, d) = 0$  if the objective value of an optimal solution is unknown. We also write  $q(G, d, \mathcal{H})$  for  $q(\mathcal{I}, \mathcal{H})$  if  $\mathcal{I} = \{DAPT(G, d)\}$ .

We evaluate also the so-called *success factor* which for a certain group of instances and a certain heuristic gives the proportion of instances for which the considered heuristic computes the best known solution.

### 6.1 Results on test instances solvable by complete enumeration

Let us first consider Table 2. All instances of this class are very small (in fact, they have only 5 – 10 vertices), and thus most of the heuristics were able to return an optimal solution for many instances. The success factors for the instances of this group are summarised in Fig. 11 (the acronyms are listed on the last page). The *DB* entry shows us the proportion of the test instances whose optimal objective function value equals the degree bound. The degree bound coincides with the optimal objective function value only in the special case  $d = n$ . Notice that in this case all arrangements yield the same objective function value and the DAPT is trivial.

### 6.2 Results on test instances solvable in polynomial time

Table 3 is related to the instances which can be solved by a polynomial time algorithm. This group of instances is divided into four parts as follows.

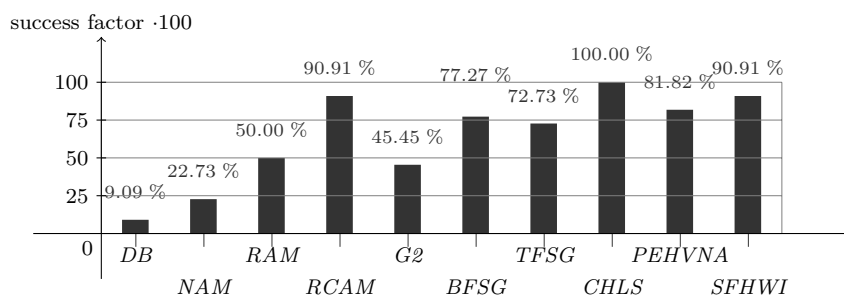


Fig. 11 Success factors for the instances solved by complete enumeration.

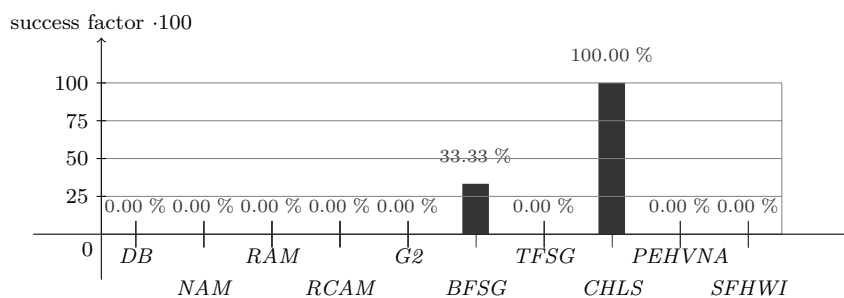


Fig. 12 Success factors for the instances with  $d = n - 1$ .

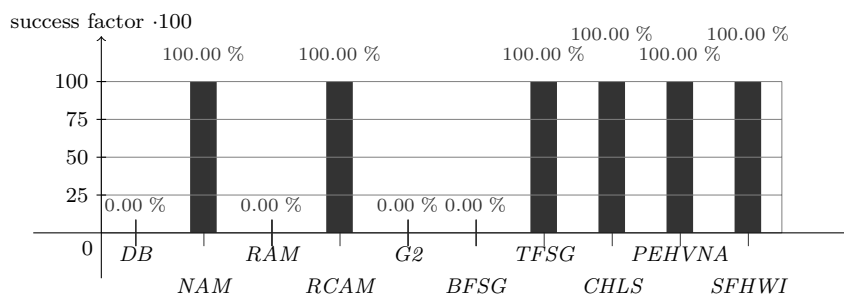
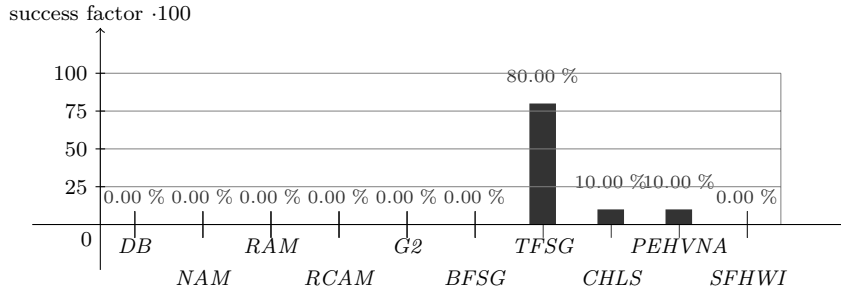


Fig. 13 Success factors for the instance with  $d = 4$  and the guest graph of Fig. 3.

- (i) Instances for which the equality  $d = n - 1$  holds. The corresponding success factors are given in Fig. 12. It is interesting that *CHLS* yields the optimal solution for any instance of this group, which of course does not hold for all such DAPT instances in general, cf. e.g. Fig. 14.
- (ii) Instances whose guest graph is a *star*, a *simple path* or a *simple cycle*. Most of the heuristics return an optimal solution.
- (iii) The instance with the guest graph of Fig. 3 and  $d = 4$ . The corresponding success factors are given in Fig. 13. Notice that neither the lower bound



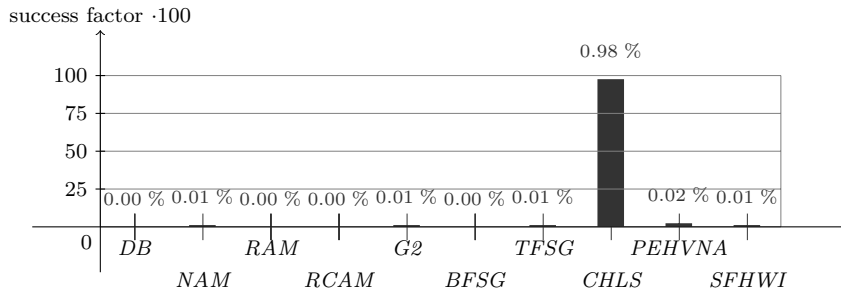
**Fig. 14** Success factors for the instances with a  $d$ -regular tree as a guest graph.

nor any heuristic is able to reach the optimum. Notice also that only some heuristics can generate a non-continuous arrangement. In our implementation they are RAM, CHLS and SFHWI.

- (iv) Instances with a  $d$ -regular tree as a guest graph. No heuristic is able to return an optimal arrangement for these instances and *TFSG* performs mostly better than *CHLS*. The quality of the solutions is  $q(\mathcal{I}, \mathcal{H}) \approx 1.18$ . The corresponding success factors are given in Fig. 14.

### 6.3 Results on test instances with unknown optimal solution

Let us now consider the test instances with unknown optimal solution, i.e. the optimal solution of this instances is not obtained by complete enumeration and it is not known whether it can be computed in polynomial time, see Table 4. The corresponding success factors are given in Fig. 15.



**Fig. 15** Success factors for the instances with an unknown optimum.

In the following we make some remarks on the particular classes of instances from this group.

Consider first the randomly generated instances (with prefix *RG*). For all these instances *CHLS* outperforms the other heuristics. We also observe



that for any fixed  $d$  the quotients  $q(\mathcal{I}, \mathcal{H})$  are better for denser graphs. The overall quality quotient for all these instances is  $q(\mathcal{I}, \mathcal{H}) \approx 1.21$ . The quality quotient is better if  $d = 2$ ; we get  $q(\mathcal{I}, \mathcal{H}) \approx 1.20$  over the instances  $\mathcal{I}$  with  $d = 2$  and  $q(\mathcal{I}, \mathcal{H}) \approx 1.23$  for the other instances of this group.

For the random instances ( $RG\_randomAx$  and  $RG\_randomBx$ ) we also observe an improvement of the quality quotient depending on the increasing expected density of the guest graph. Fig. 16 shows the values of quality quotient computed for each pair of instances with guest graphs  $RG\_randomAx$  and  $RG\_randomBx$ , for  $x \in \{5, 15, 25, 25, 45, 55, 65, 75, 85, 95\}$ , and  $d = 2$  or  $d = 7$  respectively. Clearly  $x$  represents the expected density of graphs generated as described in Section 5.3.

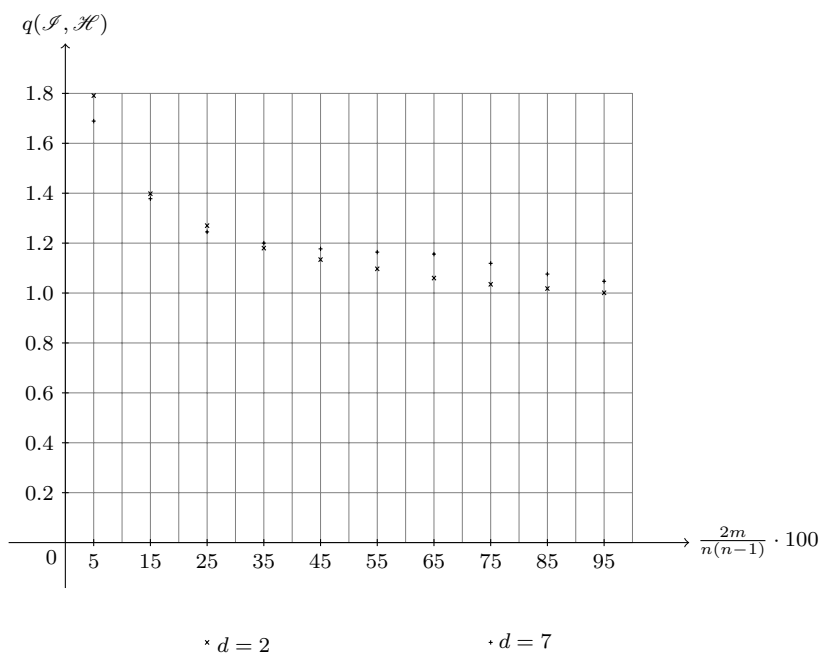
Next consider the instances with guest graphs taken from Petit (2003). Let us notice that we have not considered the guest graphs  $Pet03\_crack$  with  $d = 2$  and have also excluded  $Pet03\_wave$  and  $Pet03\_small$  as guest graphs from our tests. The reason is the big size of the guest graphs for the first two cases and the obtained solution by complete enumeration in the third case. The quality quotient is  $q(\mathcal{I}, \mathcal{H}) \approx 1.84$  for this group of instances. For  $d = 2$  we get  $q(\mathcal{I}, \mathcal{H}) \approx 1.93$  and for  $d = 7$  we get  $q(\mathcal{I}, \mathcal{H}) \approx 1.76$ . Notice that the quality quotient is worse for these instances than for the  $RG\_$  instances.

A special behaviour could be observed on following test instances:

- The guest graph is given by  $Pet03\_hc10$  and  $d = 2$ . The underlying graph corresponds to a 10-hypercube. Five heuristics yield solutions with the same objective function value which is the best know so far. It is worth of investigating whether this objective function value is optimal.
- The guest graph is given by  $Pet03\_bintree10$  (a binary tree of height 10) and  $d = 7$ . This problem is polynomially solvable in the case that  $d = 2$  (see Çela and Staněk (2013)). For  $d \neq 2$  the computational complexity of this problem is still open. We observe that  $TFSG$  performs better than  $CHLS$  for both instances with the guest graph  $Pet03\_bintree10$  and  $d = 7$  or  $d = 2$ , respectively.

#### 6.4 Performance of the construction heuristic

In Tables 2, 3 and 4 only the variant of the construction heuristic which uses the simple local search idea (see Section 4.2.1) to solve MCBSSP is included. This strategy outperforms the other one which uses the algorithm proposed by Feige et al (2003) as a subroutine to solve MCBSSP. Table 1 provides some results on the comparison of the construction heuristic involving both approaches to solve MCBSSP, respectively. In this table there is only one instance for which the involvement of the algorithm of Feige et al. yields better results. The guest graph of this instance is 2-regular tree with  $d = 2$ , hence this is an instance of a special case of the DAPT solvable in polynomial time, see Çela and Staněk (2013).



**Fig. 16** Progress of the quality quotient  $q(\mathcal{I}, \mathcal{H})$  as a function of the expected density of the randomly generated guest graphs.

Table 1 – Two strategies in our construction heuristic						
Graph	d	n	m	OS	LS	FKN
SC_random50	499	500	62468	<b>125374</b>	<b>125374</b>	<b>125374</b>
SC_treeDG2H8	2	511	510	2434	3466	<b>3188</b>
SC_treeDG3H6	3	1093	1092	3926	<b>5042</b>	5234
RG_randomA5	2	500	6126	–	<b>86628</b>	94660
RG_randomA55	2	500	68320	–	<b>1074734</b>	1091954
RG_randomA95	2	500	118499	–	<b>1893354</b>	1898482
RG_randomA5	7	500	6126	–	<b>36392</b>	39344
RG_randomA55	7	500	68320	–	<b>446038</b>	452220
RG_randomA95	7	500	118499	–	<b>785158</b>	787360
Pet03_randomA1	2	1000	4974	–	<b>71874</b>	81264
Pet03_hc10	2	1024	5120	–	<b>56320</b>	<b>56320</b>
Pet03_c1y	2	828	1749	–	<b>16884</b>	21454
Pet03_gd95c	2	62	144	–	<b>866</b>	1044
Pet03_randomA1	7	1000	4974	–	<b>29574</b>	33980
Pet03_hc10	7	1024	5120	–	<b>25892</b>	27580
follow-up on the next page ...						

Graph	d	n	m	OS	LS	FKN
Pet03_c1y	7	828	1749	–	<b>7508</b>	9016
Pet03_gd95c	7	62	144	–	<b>410</b>	500

Table 1: *A comparison of the two approaches used to solve MCBSSP as a subroutine in the construction heuristic: the local search idea (LS) and the algorithm proposed by Feige et al (2003) (FKN), see Section 4.2.1.*

## 7 Conclusions and outlook

In this paper we deal with the data arrangement problem on regular trees DAPT, identify some basic properties and introduce heuristic approaches for this problem. We provide a comparative analysis of the proposed heuristics based on a set of test instances we have generated. To the best of our knowledge no sources of literature dealing with heuristic approaches for the DAPT are available. So there is no possibility to test the performance of the proposed heuristics on already known benchmark instances and neither to compare the proposed heuristics to already existing approaches in the literature. However we make use of test instances available in Petit (2003) for a related problem, the linear arrangement problem, and use these graphs as a guest graph in our test instances. We have summarised the generated test instances in a library which is available at <http://www.opt.math.tu-graz.ac.at/~cela/public.htm> and at <http://www.rostislavstanek.at/daten/DAPTLIB.zip>.

There is plenty of room for further research on this topic in the future. Most of the heuristics we propose are basis approaches which can be well combined with one another. Especially we expect a significant performance improvement if the two local search heuristics we propose are combined in order to escape from the local minima of our neighbourhood by making a jump in the other neighbourhood. Also in the construction heuristic there is room for improvement, especially as far as the subroutine used to solve MCBSSP is concerned. Since this problem has been investigated to some extent in the literature there is hope for appropriate approaches to make use of in the construction heuristic. Another aspect which could be considered is an alternative handling of the unused leaves.

## Acknowledgements

The research was funded by the Austrian Science Fund (FWF): P23829.

## References

- ela E, Staněk R (2013) Polynomially solvable special cases of the data arrangement problem on regular trees, working paper
- Chung FRK (1984) An optimal linear arrangement of trees. *Comput and Math with Appl* 10:43–60
- Feige U, Krauthgamer R, Nissim K (2003) On cutting a few vertices from a graph. *Discret Appl Math* 127:643–649
- Garey MR, Johnson DS (1979) *Computers and intractability: a guide to the theory of NP-completeness*, Series of books in the mathematical sciences, p 210
- Juvan M, Mohar B (1992) Optimal linear labelings and eigenvalues of graphs. *Discret Appl Math* 36(2):153–168
- Luzcak MJ, Noble SD (1992) Optimal arrangement of data in a tree directory. *Discret Appl Math* 121 (1–3):307–315
- Petit J (1998) Approximation heuristics and benchmarkings for the MinLA problem. In: Battiti R, Bertossi A (eds) “Algorithms and Experiments” (ALEX98) – Building bridges between theory and applications, pp 112–128
- Petit J (2003) Experiments on the minimum linear arrangement problem. *ACM J of Exp Algorithm* 8:307–315
- Shiloach Y (1979) A minimum linear arrangement algorithm for undirected trees. *SIAM J on Comput* 8:15–22
- Staněk R (2012) Heuristiken für das optimale data-arrangement-problem in einem baum. Master’s thesis, Graz Univeristy of Technology, in German

## Appendix

### Proof of Proposition 1

We make use of Observation 1 in order to show that  $d_T(\phi(v_i), \phi(v_j)) = d_T(\phi_f(v_i), \phi_f(v_j))$  for any edge  $(v_i, v_j)$  of the guest graph  $G$ . For  $v_i \in V$ ,  $i = 1, 2, \dots, n$ , let us denote by  $p(i)$ ,  $p_f(i)$  the indices of the leaves  $\phi(v_i)$ ,  $\phi_f(v_i)$  of  $T$  in the canonical ordering, respectively. We clearly have  $p(i), p_f(i) \in 1, 2, \dots, d^h$ , for all  $i = 1, 2, \dots, n$ . According to Observation 1 we get

$$d_T(\phi(v_i), \phi(v_j)) = 2 \operatorname{argmin} \left\{ k \in \{1, 2, \dots, h\} : \left\lfloor \frac{p(i)-1}{d^k} \right\rfloor = \left\lfloor \frac{p(j)-1}{d^k} \right\rfloor \right\} \quad (19)$$

and

$$d_T(\phi_f(v_i), \phi_f(v_j)) = 2 \operatorname{argmin} \left\{ k \in \{1, 2, \dots, h\} : \left\lfloor \frac{p_f(i)-1}{d^k} \right\rfloor = \left\lfloor \frac{p_f(j)-1}{d^k} \right\rfloor \right\} \quad (20)$$

Consider the index  $p$  of an arbitrary leaf  $b_p$  of  $T$  (in the canonical order) written as  $p = (u - 1)d^{h-e} + (s - 1)d^{h-(e+1)} + t$  for some natural numbers  $1 \leq u \leq d^e$ ,  $1 \leq s \leq d$  and  $1 \leq t \leq d^{h-(e+1)}$ .  $u$  represents the index of the unique vertex  $x$  at level  $e$  which is an ancestor of  $b_p$ ,  $s$  represents the index of the  $d$ -regular subtree  $T_1$  of height  $h - (e + 1)$  hanging on  $x$  and  $t$  represents the index of  $b_p$  in  $T_1$  according to the canonical order of the leaves of  $T_1$  induced by the canonical order of the leaves of  $T$ . Then the following equality holds

$$\left\lfloor \frac{p-1}{d^k} \right\rfloor = \begin{cases} (u-1)d^{h-e-k} + (s-1)d^{h-(e+1)-k} + \left\lfloor \frac{t-1}{d^k} \right\rfloor & \text{if } k < h - (e+1) \\ (u-1) & \text{if } k = h - e \\ \left\lfloor \frac{u}{d^{k-(h-e)}} \right\rfloor & \text{if } k > h - e \\ (u-1)d + (s-1) & \text{if } k = h - (e+1) \end{cases}, \quad (21)$$

for any  $1 \leq u \leq d^e$ , any  $1 \leq s \leq d$  and any  $1 \leq t \leq d^{h-(e+1)}$ . Notice that according to Definition 4  $\phi(v_i) \neq \phi_f(v_i)$  holds, only if  $p(i) = \Delta(g) + (l-1)d^{h-(e+1)} + t_i$  or  $p(i) = \Delta(g) + (r-1)d^{h-(e+1)} + t_i$  with some  $1 \leq t_i \leq d^{h-(e+1)}$ . Moreover, the following two implications hold for  $t_i = 1, 2, \dots, d^{h-(e+1)}$ :

$$p(i) = \Delta(g) + (l-1)d^{h-(e+1)} + t_i \implies p_f(i) = \Delta(g) + (r-1)d^{h-(e+1)} + t_i, \quad (22)$$

$$p(i) = \Delta(g) + (r-1)d^{h-(e+1)} + t_i \implies p_f(i) = \Delta(g) + (l-1)d^{h-(e+1)} + t_i. \quad (23)$$

Consider now an edge  $(v_i, v_j)$  with  $\phi(v_i) \neq \phi_f(v_i)$  or  $\phi(v_j) \neq \phi_f(v_j)$ , which is equivalent to  $p(i) \neq p_f(i)$  or  $p(j) \neq p_f(j)$ . There are two cases: (I)  $p(i) \neq p_f(i)$  and  $p(j) \neq p_f(j)$ , or (II) just one of the inequalities  $p(i) \neq p_f(i)$ ,  $p(j) \neq p_f(j)$  holds.

**Case I.** In this case one of the following subcases can happen:

Case Ia.  $p(i) = \Delta(g) + (l-1)d^{h-(e+1)} + t_i$  and  $p(j) = \Delta(g) + (l-1)d^{h-(e+1)} + t_j$ ,

or

Case Ib.  $p(i) = \Delta(g) + (r-1)d^{h-(e+1)} + t_i$  and  $p(j) = \Delta(g) + (r-1)d^{h-(e+1)} + t_j$ ,

or

Case Ic.  $p(i) = \Delta(g) + (l-1)d^{h-(e+1)} + t_i$  and  $p(j) = \Delta(g) + (r-1)d^{h-(e+1)} + t_j$ ,

or

Case Id.  $p(i) = \Delta(g) + (r-1)d^{h-(e+1)} + t_i$  and  $p(j) = \Delta(g) + (l-1)d^{h-(e+1)} + t_j$ .

In Case Ic and in Case Id we get  $d(\phi(i), \phi(j)) = d(\phi_f(i), \phi_f(j)) = 2(h-e)$  by applying (21) and considering (22), (23). In Case Ia and in Case Ib we get

$$d(\phi(i), \phi(j)) = d(\phi_f(i), \phi_f(j)) =$$

$$2 \min \left\{ h - (e+1), \operatorname{argmin} \left\{ k \in \{1, 2, h - (e+2)\} : \frac{t_i - 1}{d^k} = \frac{t_j - 1}{d^k} \right\} \right\}.$$

**Case II.** Assume w.l.o.g. that  $p(i) = (g-1)d^{h-e} + (l-1)d^{h-(e+1)} + t_i$  and let  $p(j) = (u-1)d^{h-e} + (s-1)d^{h-(e+1)} + t_j$ , where  $g \neq u$  or  $s \notin \{l, r\}$ .

Clearly  $p_f(i) = (g - 1)d^{h-e} + (r - 1)d^{h-(e+1)} + t_i$  and  $p_f(j) = p(j) = (u - 1)d^{h-e} + (s - 1)d^{h-(e+1)} + t_j$ . If  $u = g$  and  $s \notin \{l, r\}$ , then (21) together with Observation 1 implies  $d_T(\phi(i)\phi(j)) = d_T(\phi_f(i)\phi_f(j)) = 2(h - e)$ .

Otherwise, if  $u \neq g$ , then (21) implies  $\left\lfloor \frac{p(i)-1}{d^k} \right\rfloor = \left\lfloor \frac{p_f(i)-1}{d^k} \right\rfloor$  for all  $k \geq h - e$  and

$$\left\lfloor \frac{p(i)-1}{d^k} \right\rfloor \neq \left\lfloor \frac{p(j)-1}{d^k} \right\rfloor ; \left\lfloor \frac{p_f(i)-1}{d^k} \right\rfloor \neq \left\lfloor \frac{p(j)-1}{d^k} \right\rfloor = \left\lfloor \frac{p_f(j)-1}{d^k} \right\rfloor , \text{ for all } k < h - e,$$

which together with Observation 1 implies then

$$d_T(\phi(i), \phi(j)) = d_T(\phi_f(i), \phi_f(j)).$$

Thus  $d_T(\phi(i)\phi(j)) = d_T(\phi_f(i)\phi_f(j))$  for any edge  $(v_i, v_j) \in E$ .

Tables of numerical results

Table 2 – summary for the instances solved by the complete enumeration

Graph	d	n	m	OS	DB	NAM	RAM	RCAM	G2	BFSG	TFSG	CHLS	PEHVNA	SFHWT
CE_sample	3	5	7	20	18	22	20	20	22	20	20	20	20	20
CE_sparse7	2	7	2	24	21	26	24	24	26	24	24	24	26	24
CE_sparse7	3	7	2	18	16	20	18	18	20	18	18	18	18	18
CE_sparse7	4	7	2	16	14	16	18	16	16	16	16	16	16	16
CE_sparse7	5	7	2	16	14	18	18	16	18	18	18	16	18	16
CE_sparse7	6	7	2	16	14	18	18	16	18	16	16	16	16	16
CE_sparse7	7	7	2	14	14	14	14	14	14	14	14	14	14	14
CE_dense7	2	7	14	62	56	64	62	64	64	62	66	62	62	62
CE_dense7	3	7	14	44	42	44	44	44	44	44	46	44	44	44
CE_dense7	4	7	14	40	35	42	40	42	40	42	40	40	40	40
CE_dense7	5	7	14	38	30	40	40	38	38	38	38	38	38	38
CE_dense7	6	7	14	34	28	36	38	34	34	34	34	34	34	34
CE_dense7	7	7	14	28	28	28	28	28	28	28	28	28	28	28
CE_mesh9	2	9	12	54	40	58	62	56	58	62	54	54	54	54
CE_mesh9	3	9	12	36	30	36	36	36	36	36	36	36	36	36
CE_mesh9	4	9	12	34	25	36	36	34	36	36	34	34	34	34
CE_sparse10	2	10	11	46	34	62	54	48	60	56	46	46	48	48
CE_sparse10	4	10	11	30	22	34	32	30	34	32	30	30	30	30
CE_dense10	2	10	26	134	118	140	150	134	136	134	140	134	134	136
CE_dense10	4	10	26	80	74	82	86	80	80	80	86	80	82	80
Pet03_small	2	5	8	34	28	36	34	34	34	34	34	34	34	34
Pet03_small	3	5	8	24	22	26	24	24	24	24	24	24	24	24

Table 2: Summary for the instances solved by the complete enumeration.

Table 3 – summary for the instances solved by a polynomial time algorithm

Graph	d	n	m	OS	DB	NAM	RAM	RCAM	G2	BFSG	TFSG	CHLS	PEHVNA	SFWHI
SC_random25	499	500	31239	<b>62644</b>	62478	62720	124774	62740	62644	<b>62644</b>	62704	<b>62644</b>	62706	62706
SC_random50	499	500	62468	<b>125374</b>	124936	125438	249544	125410	125374	125380	125418	<b>125374</b>	125420	125438
SC_random75	499	500	93548	<b>187784</b>	187096	187832	373724	187822	187784	187796	187840	<b>187784</b>	187824	187832
SC_star50	2	50	49	<b>474</b>	286	<b>474</b>	478	<b>474</b>	<b>474</b>	<b>474</b>	<b>474</b>	<b>474</b>	<b>474</b>	<b>474</b>
SC_star500	2	500	499	<b>7978</b>	4488	<b>7978</b>	7980	<b>7978</b>	<b>7978</b>	<b>7978</b>	<b>7978</b>	<b>7978</b>	<b>7978</b>	<b>7978</b>
SC_star1000	2	1000	999	<b>17954</b>	9976	<b>17954</b>	17968	<b>17954</b>	<b>17954</b>	<b>17954</b>	<b>17954</b>	<b>17954</b>	<b>17954</b>	<b>17954</b>
SC_star50	7	50	49	<b>186</b>	142	<b>186</b>	262	<b>186</b>	<b>186</b>	<b>186</b>	<b>186</b>	<b>186</b>	<b>186</b>	<b>186</b>
SC_star500	7	500	499	<b>3200</b>	2099	<b>3200</b>	3770	<b>3200</b>	<b>3200</b>	<b>3200</b>	<b>3200</b>	<b>3200</b>	<b>3200</b>	<b>3200</b>
SC_star1000	7	1000	999	<b>7200</b>	4599	<b>7200</b>	7600	<b>7200</b>	<b>7200</b>	<b>7200</b>	<b>7200</b>	<b>7200</b>	<b>7200</b>	<b>7200</b>
SC_extStar	4	12	11	28	23	<b>30</b>	32	<b>30</b>	32	<b>30</b>	<b>30</b>	<b>30</b>	<b>30</b>	<b>30</b>
SC_treeDG2H8	2	511	510	2434	1529	8176	7968	7982	8176	7680	<b>2746</b>	3466	4878	6426
Pet03_bintree10	2	1023	1022	4904	3065	18414	18146	18118	18414	17410	<b>5618</b>	7072	10696	15030
SC_treeDG2H10	2	2047	2046	9850	6137	40940	45508	40542	40940	38914	<b>11418</b>	16026	23566	34938
SC_treeDG2H11	2	4095	4094	19744	12281	90090	89568	89528	90090	86020	<b>23154</b>	33748	50826	80064
SC_treeDG2H12	2	8191	8190	39538	24569	196584	195714	195668	196584	188420	<b>46930</b>	71666	109504	174972
SC_treeDG3H5	3	364	363	1296	967	3640	3870	3642	3640	3640	1524	<b>1472</b>	2376	2778
SC_treeDG3H6	3	1093	1092	3926	2911	13116	14040	13214	13116	13116	<b>4772</b>	5042	8380	11314
SC_treeDG4H4	4	341	340	1058	849	2728	3102	2738	2728	2728	<b>1288</b>	1328	1584	2144
SC_treeDG4H5	4	1365	1364	4272	3409	1365	15310	13884	13650	13650	<b>5320</b>	5412	7626	11820
SC_treeDG8H3	8	585	584	1472	1313	3510	4426	3526	3510	3510	1956	1808	<b>1716</b>	3018
SC_path50	2	50	49	<b>190</b>	146	<b>190</b>	434	434	<b>190</b>	<b>190</b>	<b>190</b>	<b>190</b>	<b>190</b>	<b>190</b>
SC_path500	2	500	499	<b>1982</b>	1496	<b>1982</b>	7818	7814	<b>1982</b>	<b>1982</b>	<b>1982</b>	<b>1982</b>	<b>1982</b>	<b>1982</b>
SC_path1000	2	1000	999	<b>3980</b>	2996	<b>3980</b>	17706	17726	<b>3980</b>	<b>3980</b>	<b>3980</b>	<b>3980</b>	<b>3980</b>	<b>3980</b>
SC_path50	7	50	49	<b>114</b>	98	<b>114</b>	258	180	<b>114</b>	<b>114</b>	<b>114</b>	<b>114</b>	<b>114</b>	<b>114</b>
SC_path500	7	500	499	<b>1162</b>	998	<b>1162</b>	3754	3260	<b>1162</b>	<b>1162</b>	<b>1162</b>	<b>1162</b>	<b>1162</b>	<b>1162</b>
SC_path1000	7	1000	999	<b>2326</b>	1998	<b>2326</b>	7562	7114	<b>2326</b>	<b>2326</b>	<b>2326</b>	<b>2326</b>	<b>2326</b>	<b>2326</b>
SC_simpleCycle50	2	50	50	<b>202</b>	150	<b>202</b>	436	452	284	284	<b>202</b>	<b>202</b>	<b>202</b>	<b>202</b>
SC_simpleCycle500	2	500	500	<b>2000</b>	1500	<b>2000</b>	7782	7804	2968	2968	<b>2000</b>	<b>2000</b>	<b>2000</b>	<b>2000</b>
SC_simpleCycle1000	2	1000	1000	<b>4000</b>	3000	<b>4000</b>	17742	17746	5964	5964	<b>4000</b>	<b>4000</b>	<b>4000</b>	<b>4000</b>
SC_simpleCycle50	7	50	50	<b>120</b>	100	<b>120</b>	258	178	132	132	<b>120</b>	<b>120</b>	<b>120</b>	<b>120</b>
SC_simpleCycle500	7	500	500	<b>1170</b>	1000	<b>1170</b>	3770	3256	1328	1328	<b>1170</b>	<b>1170</b>	<b>1170</b>	<b>1170</b>
SC_simpleCycle1000	7	1000	1000	<b>2334</b>	2000	<b>2334</b>	7576	7128	2656	2656	<b>2334</b>	<b>2334</b>	<b>2334</b>	<b>2334</b>

Table 3: Summary for the instances solved by a polynomial time algorithm.



Table 4 – summary for the instances without a known optimal solution

Graph	d	n	m	DB	NAM	RAM	RCAM	G2	BFSG	TFSG	CHLS	PEHVNA	SFWHI
RG_randomA5	2	500	6126	48361	98140	97640	97566	92076	96816	98004	86628	89684	93230
RG_randomB5	2	500	6175	48880	99288	98334	98344	92836	97556	98610	87540	90562	94990
RG_randomA15	2	500	18654	201234	299322	298002	297740	290074	297304	297338	281686	286346	292550
RG_randomB15	2	500	18887	204529	302016	301636	301704	293428	300738	302536	285518	290026	295792
RG_randomA25	2	500	31254	379064	500784	499738	499738	499796	499032	500724	480966	486306	495964
RG_randomB25	2	500	31114	376931	497684	497456	497524	487788	496732	498384	478812	483780	493214
RG_randomA35	2	500	43605	574180	699352	697740	697594	687438	697148	698420	677870	683514	691738
RG_randomB35	2	500	43595	574020	699236	697240	697540	686808	696756	698494	677294	683192	690294
RG_randomA45	2	500	56653	782958	908042	907104	906882	896474	906354	907852	886786	892358	899844
RG_randomB45	2	500	55627	766539	891646	890022	890094	879572	889732	891178	870416	876214	884740
RG_randomA55	2	500	68320	978888	1095540	1093664	1093718	1083122	1093128	1094468	1074734	1079810	1090610
RG_randomB55	2	500	68701	985749	1101882	1100048	1099958	1089652	1099576	1101090	1080722	1085512	1094074
RG_randomA65	2	500	81279	1212022	1302766	1301848	1301356	1291892	1300882	1302240	1284086	1288564	1295348
RG_randomB65	2	500	81172	1210096	1301186	1300226	1299980	1289980	1299550	1300660	1282456	1286966	1293572
RG_randomA75	2	500	93347	1429246	1496336	1495498	1495320	1486398	1495054	1495980	1479758	1484062	1490928
RG_randomB75	2	500	93399	1430182	1497266	1496448	1495956	1487202	1496172	1497070	1480906	1484748	1492458
RG_randomA85	2	500	106047	1657846	1699742	1699524	1699104	1692306	1698948	1699578	1687470	1690196	1693914
RG_randomB85	2	500	106111	1658998	1700626	1700554	1700062	1692992	1699822	1700268	1688546	1691352	1697034
RG_randomA95	2	500	118499	1881982	1899982	1899538	1899100	1895412	1899376	1899696	1893354	1894696	1897084
RG_randomB95	2	500	118606	1883908	1900908	1901264	1900678	1897246	1900698	1900804	1895088	1896222	1898688
RG_randomA5	7	500	6126	21504	40774	46738	40522	38070	39990	40518	36392	37494	39780
RG_randomB5	7	500	6175	21700	41204	47124	40886	38358	40222	40816	36570	37794	40504
RG_randomA15	7	500	18654	84924	124204	142714	123766	119752	123254	124042	117348	118986	124010
RG_randomB15	7	500	18887	86322	125500	144464	125316	121384	124626	125386	118666	120570	125044
RG_randomA25	7	500	31254	160524	207686	239158	207686	203046	206856	207712	199806	201802	207458
RG_randomB25	7	500	31114	159684	206620	238134	206588	201778	205850	206552	198944	201050	206506
RG_randomA35	7	500	43605	234630	289994	333908	289584	284438	288968	289716	281458	283704	289994
RG_randomB35	7	500	43595	234570	290000	333834	289650	284588	288862	289652	281502	283608	290000
RG_randomA45	7	500	56653	312918	376680	433734	376316	371212	375824	376474	368306	370416	376604
RG_randomB45	7	500	55627	306762	369694	426026	369460	364486	368858	369702	361318	363534	369646
RG_randomA55	7	500	68320	382920	454306	523376	454142	448710	453224	453876	446038	448168	454306
RG_randomB55	7	500	68701	385206	456812	526276	456356	451350	456894	456454	448314	450618	456812
RG_randomA65	7	500	81279	460795	540234	622664	540098	535146	539482	540146	532718	534564	540234
RG_randomB65	7	500	81172	460159	539620	621798	539254	534380	538760	539478	532046	534046	539620
RG_randomA75	7	500	93347	548776	620442	715076	620622	616240	619842	620570	613984	615588	620442
RG_randomB75	7	500	93399	549192	621312	715644	620878	616030	620392	620938	614182	616400	621312
RG_randomA85	7	500	106047	650376	705130	812572	705052	701458	704718	704864	699976	701318	705130
RG_randomB85	7	500	106111	650888	705676	812948	705388	701914	705162	705472	700330	701820	705286

follow-up on the next page ...

Table 4 – summary for the instances without a known optimal solution – follow up

Graph	d	n	m	DB	NAM	RAM	RCAM	G2	BFSG	TFSG	CHLS	PEHVNA	SFWHI
RG_randomA95	7	500	118499	749992	788218	907730	787964	786042	787888	788060	785158	785984	788214
RG_randomB95	7	500	118606	750848	788782	908732	788638	786674	788496	788652	785996	786804	788642
Pet03_randomA1	2	1000	4974	29154	89750	88988	88944	80096	87408	86806	71874	75440	86824
Pet03_randomA2	2	1000	24738	239917	446298	444500	444384	426856	442944	445890	411242	42540	443786
Pet03_randomA3	2	1000	49820	577482	897992	895654	895760	873202	894196	897554	852854	864912	894160
Pet03_randomA4	2	1000	8177	56759	147424	146646	146528	135366	145032	146066	125374	130530	144756
Pet03_randomG4	2	1000	8173	56961	147164	146030	146536	98482	93990	96648	74282	106720	135804
Pet03_hc10	2	1024	29696	29696	56320	91468	91672	56320	88684	84266	56320	56320	56320
Pet03_mesh33x33	2	1089	2112	8320	18942	41788	38714	19350	26152	23268	18722	18900	18904
Pet03_3elt	2	4720	13722	63462	169346	328306	313178	189096	174530	220584	120332	132904	168708
Pet03_airfoil1	2	4253	12289	56732	148896	293980	273364	165388	153078	188894	106416	119024	148382
Pet03_crack	2	10240	30380	145618	726664	788288	762606	426592	393516	497116	–	441042	696122
Pet03_whitaker3	2	9800	28989	134741	375730	752132	723426	371946	355240	492390	301320	335630	375298
Pet03_big	2	15606	45878	212875	650814	1190894	1189886	726976	650746	830690	436908	482936	649748
Pet03_wave	2	156317	1059331	6884189	21067766	36008138	34921840	23977688	21016364	23484964	–	–	20711426
Pet03_c1y	2	828	1749	8609	24712	31192	30752	21392	26068	22924	16884	19846	23174
Pet03_c2y	2	980	10246	29726	37392	37392	37392	25282	30232	26722	20478	24110	11592
Pet03_c3y	2	1327	2844	13578	41996	56492	54426	35066	44664	38692	28810	33736	33736
Pet03_c4y	2	1366	2915	13529	43490	57848	56106	37034	44186	38306	27930	34124	42814
Pet03_c5y	2	1202	2577	12120	37636	50712	48414	32894	37942	33524	25572	29328	35858
Pet03-gd95c	2	62	144	643	1016	1384	1354	1080	1024	1002	866	916	920
Pet03-gd96a	2	1096	1676	7021	30310	33124	30774	23050	27908	19060	18004	19926	28526
Pet03-gd96b	2	111	193	971	2410	2246	2200	1762	1820	1768	1486	1760	1576
Pet03-gd96c	2	65	125	495	1276	1394	1236	882	936	964	824	950	844
Pet03-gd96d	2	180	228	1002	2446	3070	2952	2592	2802	2050	1822	2054	2024
Pet03_randomA1	7	1000	4974	14006	35988	37952	35680	32204	35032	35120	29574	30608	35002
Pet03_randomA2	7	1000	24738	96266	178872	189270	178334	171354	177498	178924	166598	169020	178052
Pet03_randomA3	7	1000	49820	244920	360072	381548	359372	350714	358698	359846	343664	347592	359368
Pet03_randomA4	7	1000	8177	26710	59134	62466	58842	54702	58128	58810	51290	52990	58048
Pet03_randomG4	7	1000	8173	26697	59124	62296	58840	40294	39452	40402	31838	43626	56006
Pet03_hc10	7	1024	5120	14336	26204	39034	36824	26280	35020	34452	25892	25940	26192
Pet03_mesh33x33	7	1089	2112	4224	8294	16044	15298	8326	10478	10076	8224	8262	8286
Pet03_bintree10	7	1023	1022	2044	7384	7752	7288	7384	6488	2856	3030	4418	6436
Pet03_3elt	7	4720	13722	27694	68636	132340	120844	73754	71042	86290	52420	55328	68576
Pet03_airfoil1	7	4253	12289	24792	60750	118422	107822	64224	63170	78482	46708	49868	60466
Pet03_crack	7	10240	30380	69741	277928	293238	287434	168802	158460	193856	138178	175252	269060
Pet03_whitaker3	7	9800	28989	58482	154188	279798	273056	153490	147924	191430	126198	137200	153272
Pet03_big	7	15606	45878	92511	261418	442852	442412	286180	264862	317434	186158	199324	260680

follow-up on the next page ...

Table 4 – summary for the instances without a known optimal solution – follow up

Graph	d	n	m	DB	NAM	RAM	RCAM	G2	BFSG	TFSG	CHLS	PEHVNA	SFWI
Pet03_wave	7	156317	1059331	3299657	8223278	14474822	13238898	9203558	8150052	9050366	—	—	8110512
Pet03_c1y	7	828	1749	4075	10224	13294	12326	8778	10426	9454	<b>7508</b>	8290	9652
Pet03_c2y	7	980	2102	4822	12218	16010	15044	10540	12132	11058	<b>8724</b>	9986	11592
Pet03_c3y	7	1327	2844	6436	16810	21678	20930	14652	17786	15230	<b>12216</b>	13676	16612
Pet03_c4y	7	1366	2915	6442	17168	22212	21438	14884	17320	15490	<b>12460</b>	13732	16872
Pet03_c5y	7	1202	2577	5750	15112	19450	18730	13452	15170	13618	<b>10946</b>	11978	14520
Pet03_gd95c	7	62	144	320	474	780	608	446	460	492	410	<b>404</b>	450
Pet03_gd96a	7	1096	1676	3800	12056	12736	12092	9514	11106	7990	<b>7768</b>	8308	11050
Pet03_gd96b	7	111	193	491	1030	1062	928	744	716	772	<b>682</b>	788	696
Pet03_gd96c	7	65	125	250	574	680	542	426	448	448	<b>378</b>	438	392
Pet03_gd96d	7	180	228	555	1028	1258	1188	1060	1126	876	<b>838</b>	886	958

Table 4. Summary for the instances without a known optimal solution.

## List of acronyms

- OS = optimal solution (if known).
- DB = degree bound.
- NAM = normal arrangement. The vertices  $\{v_1, v_2, \dots, v_n\}$  of the guest graph are mapped to the leaves of the  $d$ -regular tree in their canonical ordering, i.e. by  $\phi(v_i) = b_i$ , for  $i = 1, 2, \dots, n$ .
- RAM = random arrangement for  $k = 1000$ .  $k$  random mappings of the vertices of the guest graph into the leaves of the  $d$ -regular tree are constructed, their objective function values are computed, and the random mapping with the best objective function value is selected.
- RCAM = random contiguous arrangement for  $k = 1000$ .  $k$  random contiguous mappings of the vertices of the guest graph into the leaves of the  $d$ -regular tree are constructed, their objective function values are computed, and the random mapping with the best objective function value is selected.
- G2 = arrangement produced by the leaf-driven greedy heuristic, see Section 4.1.
- BFSG = arrangement produced by the breadth-first search based greedy heuristics which tries each vertex as the starting vertex, see Section 4.1. If the graph has more than one connected components, they are arranged in a random order.
- TFSG = arrangement produced by the depth-first search based greedy heuristics which tries each vertex as the starting vertex, see Section 4.1. If the graph has more than one connected components, they are arranged in a random order.
- CHLS = arrangement produced by the construction heuristic which uses the local search approach to solve the MCBSSP, see Section 4.2.
- PEHVNA = arrangement produced by the pair-exchange heuristic for vertices which starts with the normal arrangement, see Section 4.3.1.
- SFHWI = arrangement produced by the shift-flip heuristic which accepts non-improving shifts, see Section 4.3.2. The algorithm terminates if no improvement is reached after 3 days of running time.
- – = the solution could not be found in a reasonable amount of time.