Ordering a sparse graph to minimize the sum of right ends of edges

Ordering a Sparse Graph to Minimize the Sum of Right Ends of Edges

Peter Damaschke[1,2(✉)]

[1] Department of Computer Science and Engineering, Chalmers University, 41296 Göteborg, Sweden
ptr@chalmers.se
[2] Fraunhofer-Chalmers Research Centre for Industrial Mathematics, 41288 Göteborg, Sweden

Abstract. Motivated by a warehouse logistics problem we study mappings of the vertices of a graph onto prescribed points on the real line that minimize the sum (or equivalently, the average) of the coordinates of the right ends of all edges. We focus on graphs whose edge numbers do not exceed the vertex numbers too much, that is, graphs with few cycles. Intuitively, dense subgraphs should be placed early in the ordering, in order to finish many edges soon. However, our main "calculation trick" is to compare the objective function with the case when (almost) every vertex is the right end of exactly one edge. The deviations from this case are described by "charges" that can form "dipoles". This reformulation enables us to derive polynomial algorithms and NP-completeness results for relevant special cases, and FPT results.

Keywords: Minimum linear arrangement · Pick-by-order · Cycle · Tree · Dynamic programming on subsets · Elimination ordering · 2-core · 3-core

1 Introduction

We study the following problem on undirected graphs $G = (V, E)$. Our graphs may contain parallel edges and loops (and any number of loops may be attached to a vertex), but no isolated vertices. A loop at a vertex v may be formally seen as an edge vv.

MinSumEnds

Given: (1) an undirected graph $G = (V, E)$ with n vertices, and (2) n numbers $s_1 < \ldots < s_n$.

Find: A labeling, that is, a bijective mapping λ of V onto $\{s_1, \ldots, s_n\}$ that minimizes $\sum_{e \in E} \mu(e)$, where $\mu(uv) := \max\{\lambda(u), \lambda(v)\}$ for every edge $e = uv$.

We call such a labeling *optimal*, with respect to this objective function. Our objective function can be rephrased as follows. Let $L(k)$ be the number of edges

L. Gąsieniec et al. (Eds.): IWOCA 2020, LNCS 12126, pp. 224–236, 2020.
https://doi.org/10.1007/978-3-030-48966-3_17

uv such that v is the vertex with label s_k, and the label of u is smaller than or equal to k. (This includes possible loops vv.) Then the sum of edge labels is obviously $\sum_{k=1}^{n} s_k \cdot L(k)$. Informally, $L(k)$ is the "left degree" of the vertex at position k, if the vertices are placed on the number line according to their labels. This way, a labeling can be viewed as a linear ordering of the vertices, placed on points with the coordinates $s_1 < \ldots < s_n$. We may use the words labeling and ordering interchangeably.

If the labels are *equidistant*, we can without loss of generality assume that $s_k = k$ for all k.

MinSumEnds is similar to the well-known linear arrangement problem where we want to minimize the sum of edge lengths (i.e., differences of labels). Like the linear arrangement problem it can be solved straightforwardly in $O^*(2^n)$ time[1] by dynamic programming on subsets [4].

MinSumEnds can be generalized to hypergraphs. This work was directly inspired by a real-world problem: Items are stored in a shelf in a warehouse, and certain subsets of items are frequently requested. They must be fetched from the shelf, thereby walking from the left end to the place of the rightmost requested item and back. Given a set of data on the frequently requested subsets, the problem is to store the items so as to minimize the average walking distance.

The minimum linear arrangement problem is a classic NP-complete problem [12] and has been intensively studied. Approximation algorithms and inapproximability results are known [1,2,8,14], as well as exact exponential and parameterized algorithms [3,9–11], and efficient algorithms for special graph classes [5–7,13]. MinSumEnds is much less explored. In [4], the problem is called the product location problem with a single rack and a front end depot. The problem is proved to be strongly NP-complete for equidistant labels, by a reduction from the linear arrangement problem. In fact, the reduction produces graphs with possible loops, but no hypergraphs.

In the present paper we focus on graphs with barely more edges than vertices. In the warehouse application this corresponds to the rather practical case that, typically, only single items or pairs of items are requested, and the requests are not very diverse, that is, only a small number of different pairs occurs. This easily leads to graphs whose connected components are trees or have only a few cycles. Still these graphs are rather special, but this study may serve as a first step in understanding which graph properties make the problem easy or hard. Also, the related linear arrangement problem is nontrivial even for trees [7], and now we continue this line of research for MinSumEnds.

In Sect. 2 we solve MinSumEnds for some simple graphs that contain a chain of densest subgraphs. These are induced subgraphs having the maximum number of edges, given a number of vertices. In Sect. 3 we rephrase MinSumEnds in terms of so called charges and dipoles which measure the difference to an optimal labeling of a tree. Using these concepts, we eliminate vertices of degree 1, provided that the labels are equidistant; see Sects. 3 and 4. Similarly, in Sect. 5

[1] We adopt the O^* notation that focuses on the exponential terms and suppresses polynomial factors.

we eliminate connected components that are merely cycles, and we show NP-completeness of MinSumEnds for general labels, but for a graph class as "trivial" as disjoint unions of cycles. In Sect. 6 we derive an FPT algorithm in the parameter $m - n$, the number of edges minus the number of vertices (after the previously described eliminations). Here the 2-cores and 3-cores of graphs play a prominent role. We think that the structural properties shown can be useful in their own right, not only as a preparation of the FPT result that is applicable only to graphs "slightly exceeding" forests.

2 Nested Densest Subgraphs

Consider an optimal labeling λ of G and a positive integer $k \leq n$. Let G_k be the subgraph of G induced by the vertices with the k smallest labels. Then the labeling induced by λ on G_k is also an optimal labeling of G_k. This is evident by an exchange argument; note that a permutation of the first k vertices does not affect the values $L(j)$ for $j > k$. In other words: Once we have decided on the vertices that receive the labels larger than s_k, it remains to solve the MinSumEnds problem on G_k. In such a situation we say that we have *eliminated* the other vertices.

An induced subgraph H of G with k vertices is called a *densest subgraph* if H has a maximum number of edges among all induced subgraphs of G with k vertices. We say that a labeling produces *nested densest subgraphs* if, for every k, G_k is a densest subgraph.

Not every graph allows nested densest subgraphs. The smallest counterexample has three vertices u, v, w, where u and v are joined by two parallel edges, and w has one loop. Then the only densest subgraphs with $k = 1$ and $k = 2$ are induced by $\{w\}$ and by $\{u, v\}$, respectively. However, for graphs that do have nested densest subgraphs, we can characterize optimal solutions of MinSumEnds:

Lemma 1. *Any labeling that produces nested densest subgraphs is an optimal labeling.*

Proof. Consider a labeling λ produced by nested densest subgraphs, with objective value $L = \sum_{k=1}^{n} s_k \cdot L(k)$, and assume that there is a better labeling with similarly defined values $L' = \sum_{k=1}^{n} s_k \cdot L'(k)$, where $L' < L$. Since $\sum_{j=1}^{n} L'(j) = \sum_{j=1}^{n} L(j)$, this would be possible only if there were some k with $L'(k) > L(k)$. Specifically, let k be the smallest such index. Then we have $\sum_{j=1}^{k} L'(j) > \sum_{j=1}^{k} L(j)$. But this contradicts the assumption that G_k (in λ) was already a densest subgraph. □

The converse (every optimal labeling of such graphs produces nested densest subgraphs) also holds true, but we will only use the direction given in Lemma 1. Perhaps the simplest application is the case of trees.

For clarity we remark that loops as well as pairs of parallel edges count as cycles. A *forest* is a graph without cycles. Hence, in particular, a forest must not

contain loops and parallel edges. A *tree* is a connected forest. Every subgraph of a tree is, of course, a forest.

Theorem 1. MinSumEnds *is solvable in linear time on forests.*

Proof. Every tree possesses nested densest subgraphs: We can start with an arbitrary vertex and successively add a vertex that has a neighbor among the previously selected vertices. In this way, for every k, G_k has exactly $k-1$ edges, which is indeed maximal for subgraphs of a tree.

More generally, every forest possesses nested densest subgraphs: We sort the connected components (which are trees) by decreasing sizes, order the vertices in every tree as described above, and concatenate these orderings of the trees. Then every G_k has exactly $k-c(k)$ edges, where $c(k)$ is the number of connected components of G_k. It is easy to see that sorting the trees by decreasing sizes minimizes all $c(k)$, and thus all G_k in this labeling are indeed densest subgraphs. Clearly, the procedure can be implemented to run in linear time, where the sorting is done by bucketsort. □

A slightly larger graph class can still be managed in this way:

Theorem 2. MinSumEnds *is solvable in linear time on graphs with at most one cycle.*

Proof. Forests are settled by Theorem 1, hence we can suppose that the input graph G has exactly one cycle. Let k denote its length (where $k = 1$ if the cycle is a loop, and $k = 2$ if the cycle consists of two parallel edges).

First consider the case when G is connected and has exactly one cycle. Then the only densest subgraphs of G are the following: all subgraphs of $j < k$ vertices being trees, and all connected subgraphs of $j \geq k$ vertices including the cycle. Hence G has nested densest subgraphs: Starting at any vertex of the cycle, assign the k lowest labels to the vertices of the cycle in their natural ordering, and then successively assign the next label to any vertex that has a neighbor among the already labeled vertices.

The case when G is not connected is solved by combining the previous observations (also from Theorem 1). We skip the straightforward verification of the following claims.

The only densest subgraphs of G are now the following: all subgraphs of $j < k$ vertices being trees, and all subgraphs of $j \geq k$ vertices that include the cycle and intersect the smallest possible number of other connected components, where every such intersection is a subtree of the respective component.

This yields some optimal labeling in linear time: Starting at any vertex of the cycle, assign the k lowest labels to the vertices of the cycle in their natural ordering, then successively assign the next label to any vertex that has a neighbor among the already labeled vertices, until the entire connected component containing the cycle is labeled, and finally append optimal orderings of the other connected components (which are trees), sorted by decreasing sizes. □

As the above example suggests, graphs with several cycles, in general, do not have nested densest subgraphs, and we must combine the idea with other methods, in order to solve instances of MinSumEnds.

3 Charges and Dipoles

For reasons that will become apparent soon, we work from now on with the numbers $M(k) := L(k) - 1$. Obviously, minimizing $\sum_{k=1}^{n} s_k \cdot L(k)$ is equivalent to minimizing $M := \sum_{k=1}^{n} s_k \cdot M(k)$. When v is the vertex with label s_k, we may also write $M(v)$ instead of $M(k)$. Note that the prefix sum $\sum_{j=1}^{k} M(j)$ equals the number of edges in G_k minus k, and that $M(k) \geq -1$ for every k, by definition, and vertices with $M(k) = 0$ can be ignored.

If $M(k) = -1$, then we imagine a *negative charge* at point s_k on the number line. Similarly, if $M(k) > 0$, then we imagine $M(k)$ *positive charges* at point s_k. Equivalently we may imagine that the vertices (rather than the points s_k) are charged.

Next we may pair up some of these charges to *dipoles* according to the following rules. Every dipole consists of a negative charge and a positive charge of a vertex with a higher label, and every charge belongs to at most one dipole. Of course, this pairing is by no means uniquely determined. The *length* of a dipole is defined to be the absolute value of the difference of the labels at the two involved vertices. Hence every dipole contributes exactly its length to the sum M. (It may be fun to notice that the paired-up positive and negative charges "attract each other", in the sense that we want to minimize their distances.)

A labeling such that $\sum_{j=k}^{n} M(j) \geq 0$ holds for all k is said to have the *dipole property*. This is equivalent to the property that we can form dipoles that contain all negative charges. Some surplus positive charges remain outside these dipoles.

For brevity, a *tree component* is a connected component being a tree.

Lemma 2. *Every labeling of any graph without a tree component has the dipole property.*

Proof. We show the contraposition: If some labeling of a graph G fails to have the dipole property, then G has a tree component.

Hence, assume that $\sum_{j=k}^{n} M(j) < 0$ for some k, and specifically, let k be the largest such index. Then we have $M(k) = -1$, and all charges above k can be paired up to dipoles. Let H denote the subgraph of G induced by the vertices with labels $s_k, \ldots, s_n$. Since the sum of all $M(j)$ in H is negative, there also exists some connected component T of H with more negative than positive charges. But this is possible only if T is a tree, and furthermore, no edges exist between vertices of T and vertices outside H. Hence the tree T is a connected component of G as well. □

We say that two connected components C and D are *separated* in a labeling if all labels in C are smaller than all labels in D, or vice versa.

For any optimal labeling of G, trivially, the labeling restricted to any connected component T of G must be optimal, too. In particular, if T is a tree component, we can without loss of generality assume that T is labeled as in Theorem 1. Hence T contains only one charge which is negative and sits at the vertex with the lowest label in T.

Proposition 1. *Let G be a graph with a total number $t > 0$ of vertices in its tree components. For equidistant labels, there exists an optimal labeling of G where the vertices of the tree components have the t highest labels, the tree components are separated, and they are sorted by decreasing sizes.*

Proof. Given a labeling, we divide the vertex set of G in two sets X and Y consisting of the $|X|$ vertices with the lowest labels and the $|Y|$ vertices with the highest labels, respectively, where Y is a union of tree components. (Y may be empty.) Assume that not yet all tree components are in Y. Then, let $r \in X$ be the unique vertex that has a negative charge, belongs to some tree component T, and has the highest label among all such vertices.

By Lemma 2 and the assumed labeling of tree components, it follows that all charged vertices in X with higher labels than r are positively charged or belong to dipoles. Now we relabel X such that the orderings in both T and $X - T$ are preserved, but the vertices of T receive the highest labels in X. This has the following effects. The tree T is removed from X and included in Y, the negatively charged vertex r gets a higher label, and the labels of positively charged vertices as well as the lengths of the dipoles in $X - T$ can only decrease. Altogether, the objective M cannot get worse. By an inductive argument we achieve a labeling where all tree components are at the end of the ordering and are separated.

Finally, in an optimal labeling, the tree components must also be sorted by decreasing sizes as in Theorem 1. □

For the proof it is crucial that the labels are equidistant. In the case of general labels, a dipole moving to points with other coordinates can get longer, although the number of vertices between the two charges does not increase. (It is easy to produce such counterexamples.) Of course, this cannot happen if the labels are equidistant. Moreover, since the dipoles in X can only move to smaller labels, it would be sufficient to suppose labels with monotone non-decreasing distances $s_{j+1} - s_j$. However, we stick to equidistant labels, which is a more natural assumption in the warehouse application.

We have shown that, in the case of equidistant labels, by Theorems 1 and 2 we can *eliminate* all tree components (see the beginning of Sect. 2). Therefore, from now on we can focus on graphs where every connected component has at least one cycle.

4 Eliminating the Leaves

A *leaf* is a vertex of degree 1. In the case of equidistant labels we can eliminate leaves also from connected components with cycles:

Proposition 2. *Let v be a leaf in a graph without tree components. For equidistant labels, there exists an optimal labeling where v has the highest label.*

Proof. Let u denote the unique neighbor of v, and let $G - v$ denote the graph G without v and the edge uv. Since G has no tree component, neither has $G - v$. We consider any labeling where v has not the highest label.

If the label of v is larger than the label of u, then the charges of vertices in $G - v$ are identical to their charges in G. Hence, due to Lemma 2, the labeling induced on $G - v$ has the dipole property. The leaf v is not charged. Now we simply assign the highest label to v and relabel the vertices of $G - v$ preserving their ordering. This can only decrease the lengths of dipoles (since the labels are equidistant) and the labels of the positively charged vertices outside the dipoles, thus M can only decrease.

The case when the label of v is smaller than the label of u is only slightly more complicated. If $M(u) > 0$, then we put one positive charge at u aside and form a dipole on the edge uv, together with the existing negative charge at v. If $M(v) = 0$, then we create a pair of a negative and a positive charge at v, and again, we form a dipole on the edge uv, whereas the new negative charge is assigned to u. In all cases, the charges not involved in the dipole on uv are identical to those in $G - v$, and these manipulations do not alter M. Precisely as above, we assign the highest label to v and relabel the vertices of $G - v$ preserving their ordering. The dipole at uv disappears, and for the same reasons as above, M can only decrease. □

Using Proposition 2 we can eliminate any one leaf v, and the problem of optimally labeling $G - v$ remains. Of course, we can apply this step successively, until the residual graph has no leaves anymore. Therefore, from now on we can focus on graphs with minimum degree 2.

5 Eliminating and Separating the Cycle Components

A *cycle component* is a connected component which is merely a cycle. Our next observation is quite similar to Proposition 1. First we can optimally label every cycle component independently: An optimal labeling of a cycle was already observed in the proof of Theorem 2. It has one negative and one positive charge, at the vertex with the lowest and highest label, respectively. We declare them a dipole.

Proposition 3. *Let G be a graph with minimum vertex degree 2, and with a total number $c > 0$ of vertices in its cycle components. For equidistant labels, there exists an optimal labeling of G where the vertices of the cycle components have the c highest labels, and the cycle components are separated.*

Proof. Given a labeling, we divide the vertex set of G in two sets X and Y consisting of the $|X|$ vertices with the lowest labels and the $|Y|$ vertices with the highest labels, respectively, where Y is a union of cycle components. (Y may be

empty.) Assume that not yet all cycle components are in Y. Then, let C be any cycle component that is not yet in Y.

Due to the minimum degree 2, the graph G has no tree components. By Lemma 2 it follows that all charged vertices in $X - C$ are positively charged or belong to dipoles.

Now we relabel X such that the orderings in both C and $X - C$ are preserved, but the vertices of C receive the highest labels in X. This has the following effects. The cycle C is removed from X and included in Y, and the labels of positively charged vertices in $X - C$ as well as the lengths of the dipoles in both $X - C$ and in C can only decrease. Altogether, the objective M cannot get worse. By an inductive argument we achieve a labeling as described in the statement. □

Due to Proposition 3, we can also eliminate cycle components, in the case of equidistant labels. Moreover, the ordering of cycles is irrelevant, since every cycle contributes exactly its length minus 1 to M, regardless of its position in the ordering. Without further ado this settles MinSumEnds for a larger graph class than in Theorem 2, however for equidistant labels only.

Theorem 3. MinSumEnds *with equidistant labels is solvable in linear time on graphs where every connected component has at most one cycle.*

Proof. First eliminate the tree components due to Proposition 1 and the leaves due to Proposition 2, then concatenate optimal labelings of the cycles, where the permutation of the cycles is arbitrary. □

As we already observed, this approach fails for general labels; we cannot even eliminate the leaves. But let us still consider disjoint unions of cycles for a moment. This is a too special case for applications, but the interesting point is that, with the help of dipoles, we get a rather straightforward NP-completeness proof for MinSumEnds in this very special case, by a reduction from the strongly NP-complete 3-Partition problem. We stress that this reduction does not work for equidistant labels, and the result complements NP-completeness for equidistant labels but general graphs [4].

Theorem 4. MinSumEnds *is NP-complete even for disjoint unions of cycles.*

Proof. We first observe again that every cycle must be optimally labeled, and its lowest and highest labeled vertex form a dipole. Furthermore, we can separate any two cycles that are not yet separated, because this decreases the total length of the dipoles. (We stress that this holds for arbitrary labels.) Hence, an optimal labeling is given by some permutation of the cycles also here.

Let $\{x_1, \ldots, x_{3t}\}$ be an instance of 3-Partition, that is, a multiset of $3t$ positive integers. The problem asks to partition this multiset into t triples, each with the same sum that we denote q. We create $3t$ disjoint cycles of lengths $q + x_i$ $(i = 1, \ldots, 3t)$. On the number line we place t disjoint segments, each of length $4q$. In every segment we mark the $4q$ integer points. Let the gap between any two segments be larger than 1 (but otherwise arbitrary). The coordinates of the

marked integer points are our $4qt$ labels. The constructed cycles have together $3tq + tq = 4tq$ vertices.

As stated above, there exists an optimal labeling where all cycles are separated and, moreover, every cycle has a dipole with a positive and negative charge at the vertex with lowest and highest label, respectively. The total length of the dipoles is $(4q - 1)t$ if and only if we can embed every cycle entirely in some segment. Since every cycle has a length larger than q, only 3 cycles fit in every segment. Finally, in order to embed all $3t$ cycles in the 3 segments, we must divide them into t triples, each with a total of $4q = 3q + q$ vertices. This establishes the equivalence of the problem instances. □

6 Paths of Degree-2 Vertices and Cores

In the following, let G be a graph with minimum vertex degree 2 (with the understanding that every loop contributes 1 to the degree of its vertex) and without cycle components. (Recall that a cycle component is a cycle without further edges, both inside and to the rest of G.)

We call every vertex of degree larger than 2 a *principal vertex*. We call a path a *principal path* if it ends in two principal vertices (which may be identical), it has at least one inner vertex, and all its inner vertices are of degree 2. Hence the edge set of graph G can be uniquely partitioned into the edge sets of its principal paths and single edges that do not belong to principal paths as they end in two principal vertices.

Lemma 3. *Let G be a graph of minimum degree 2 and without cycle components. For equidistant labels, there exists an optimal labeling of G where either (1) some principal vertex gets the highest label, or (2) some inner vertex v of some principal path P gets the highest label, followed by all other inner vertices of P getting the next smaller labels. Furthermore, in case (2) and for any fixed P, the choice of v from P is arbitrary.*

Proof. The distinction of cases (1) and (2) is trivial, since other types of vertices do not exist in G. In case (2), where we first eliminate a vertex v from a principal path P, we can apply Proposition 2 repeatedly until the rest of P is eliminated, too. Not only the leaves may be eliminated in any order, it is also immaterial which inner vertex v from P we choose first: In any case, v gets one positive charge, and the other inner vertices of P get no charge, hence the choice of v on P does not affect the objective value M. □

Lemma 3 enables dynamic programming on subsets of principal vertices and paths (rather than just vertices):

Theorem 5. MinSumEnds *with equidistant labels can be solved in $O^*(2^p)$ time, where p is the total number of principal vertices and paths after the elimination of tree components, leaves, and cycle components.*

Proof. We eliminate principal vertices and paths as in Lemma 3, in all possible ways, but: Among all partial solutions that assign the labels larger than s_k to the same $n-k$ vertices (that is, retain the same graph G_k), it suffices to keep some solution with minimum $\sum_{i=k+1}^{n} s_k \cdot M(k)$. Furthermore, whenever we eliminate some principal vertex being incident to some principal paths, we next eliminate these paths, leaf by leaf, as in Proposition 2.

Let us call two vertices equivalent if they are inner vertices of the same principal path. That is, every principal path becomes an equivalence class. Every principal vertex is an equivalence class of its own. With this definition we observe that, during the elimination process, equivalence classes are either removed completely or they get merged, but they are never torn apart. This implies that the parameter value p never increases, and the time bound follows. □

In the following we strengthen Theorem 5 by making the parameter smaller. The next lemmas presume the same type of graphs as before.

Lemma 4. *Let P be some principal path that ends in some principal vertex v (and in some other principal vertex different from v). Then, instead of eliminating v, one can always eliminate the inner vertices of P first, without making the labeling worse.*

Proof. Let u be an arbitrary inner vertex of P, and let $d \geq 3$ denote the degree of v. If we first eliminate v, followed by P, then v receives $d-1$ positive charges. In fact, we can assume that P is completely eliminated next, as the ordering of eliminating leaves is arbitrary.

If we instead eliminate u first, followed by the rest of P and by v, then we eliminate the same set of vertices and edges as before, until that moment, but u receives only one positive charge, whereas v receives only $d-2$ positive charges which are located at smaller labels. This makes M strictly smaller, hence it is never advantageous to assign the highest label to v. □

Lemma 5. *Let P be some principal path with principal vertex v at both ends. Then, instead of eliminating v, one can always eliminate P first, without making the labeling worse.*

Proof. The argument is similar. Let u be an arbitrary inner vertex of P, and let $d \geq 3$ denote the degree of v. If we first eliminate v, followed by P, then v receives $d-1$ positive charges. Now P becomes a tree component and receives one negative charge at its lowest label.

If we instead eliminate u first, followed by the rest of P and by v, then u receives one positive charge, and v receives $d-3$ positive charges, making M strictly smaller. Hence it is not advantageous to give the highest label to v. □

Lemmas 4 and 5 together state that a principal vertex needs to be considered for elimination only if all its neighbors are principal vertices, too. Some of our results can now be nicely expressed using the notion of a core.

For any positive integer d, the *d-core* of the graph G is the graph obtained from G by removing vertices of degree smaller than d, and their incident edges,

as long as possible. The result does not depend on the order of removals. Equivalently, the d-core is the uniquely determined largest induced subgraph of G with minimum vertex degree d. Remember that we adopt the convention that a loop at a vertex v contributes only 1 to the degree of v.

Propositions 1 and 2 immediately imply:

Proposition 4. *For every graph G and for equidistant labels, there exists an optimal labeling of G where all vertices in the 2-core of G have smaller labels than all other vertices.* □

A similar statement is not true for the 3-core. A small counterexample is the graph consisting of one vertex with two loops and a clique of four vertices. The clique is the 3-core, but the only optimal labeling gives the lowest label to the two-loop vertex. However, we can somewhat strengthen Theorem 5 using the 3-core. The following parameter q is smaller than p from Theorem 5, because it includes only principal vertices in the 3-core.

Theorem 6. MinSumEnds *with equidistant labels can be solved in $O^*(2^q)$ time, where q is the number of vertices in the 3-core plus the number of principal paths, after the elimination of tree components, leaves, and cycle components.*

Proof. We proceed as in Theorem 5, but according to Lemmas 4 and 5 we never have to eliminate principal vertices outside the 3-core. □

This also implies a bound in a more natural parameter:

Theorem 7. MinSumEnds *with equidistant labels can be solved in $O^*(6^{m-n})$ time, where m and n denotes the number of edges and vertices, respectively.*

Proof. Due to Theorem 6 it suffices to show $q \leq 3(m-n)$.

We can replace every principal path of arbitrary length with a principal path with only one inner vertex, as this changes neither q nor $m-n$. Now every vertex in the 3-core and every inner vertex of a principal path contributes a summand exactly 1 to q, by the definition of q. We also divide edges with two different ends between these two vertices and thus assign fractions of edges to vertices, such that no fraction is erroneously counted twice.

Every principal vertex outside the 3-core contributes zero to q, by the definition of q. We assign $1/3$ of every incident edge to it, hence it contributes a summand at least $3 \cdot (1/3) - 1 \geq 0$ to $m - n$, that is, it does not contribute negatively. Every vertex on a principal path contributes a summand at least $2 \cdot (2/3) - 1 = 1/3$ to $m - n$ via its 2 incident edges. (In the worst case, both ends may be principal vertices that do not belong to the 3-core.) Every vertex in the 3-core contributes a summand at least $1/2 = 3/2 - 1$ to $m - n$, via halves of 3 of its incident edges within the 3-core.

In conclusion, the ratio $(m-n)/q$ is at least $1/3$. □

7 Conclusions

We considered a product location problem in warehouses, with a collection point at the end of a shelf, and with a small number of different requests of at most two items, leading to a labeling problem on sparse graphs. We believe that the FPT results can be further improved: The worst case in Theorem 7 is 3-regular graphs with subdivided edges. Then, eliminations of the principal paths cause mergings of many other principal paths, hence by far not all subsets of principal paths can appear. Also, more can be done for non-equidistant labels, weighted (instead of multiple) edges, and hypergraphs.

Acknowledgments. This work has been done during the author's engagement as scientific advisor at the Fraunhofer-Chalmers Research Centre for Industrial Mathematics, Göteborg (FCC). The author appreciates support from FCC and many discussions with Fredrik Ekstedt and Raad Salman who brought up this type of problems. He also thanks the referees for very careful reading.

References

1. Ambühl, C., Mastrolilli, M., Svensson, O.: Inapproximability results for maximum edge biclique, minimum linear arrangement, and sparsest cut. SIAM J. Comput. **40**, 567–596 (2011)
2. Arora, S., Frieze, A., Kaplan, H.: A new rounding procedure for the assignment problem with applications to dense graphs arrangements. Math. Program. **92**, 1–36 (2002)
3. Bhasker, J., Sahni, S.: Optimal linear arrangement of circuit components. In: HICSS 1987, vol. 2, pp. 99–111 (1987)
4. Boysen, N., Stephan, K.: The deterministic product location problem under a pick-by-order policy. Discrete Appl. Math. **161**, 2862–2875 (2013)
5. Cohen, J., Fomin, F., Heggernes, P., Kratsch, D., Kucherov, G.: Optimal linear arrangement of interval graphs. In: Královič, R., Urzyczyn, P. (eds.) MFCS 2006. LNCS, vol. 4162, pp. 267–279. Springer, Heidelberg (2006). https://doi.org/10.1007/11821069_24
6. Eikel, M., Scheideler, C., Setzer, A.: Minimum linear arrangement of series-parallel graphs. In: Bampis, E., Svensson, O. (eds.) WAOA 2014. LNCS, vol. 8952, pp. 168–180. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-18263-6_15
7. Esteban, J.L., Ferrer-i-Cancho, R.: A correction on shiloach's algorithm for minimum linear arrangement of trees. SIAM J. Comput. **46**, 1146–1151 (2017)
8. Feige, U., Lee, J.R.: An improved approximation ratio for the minimum linear arrangement problem. Inf. Process. Lett. **101**, 26–29 (2007)
9. Fellows, M.R., Hermelin, D., Rosamond, F.A., Shachnai, H.: Tractable parameterizations for the minimum linear arrangement problem. ACM Trans. Comput. Theory **8**, 6:1–6:12 (2016)
10. Fernau, H.: Parameterized algorithmics for linear arrangement problems. Discrete Appl. Math. **156**, 3166–3177 (2008)
11. Fomin, F.V., Kratsch, D.: Split and list. In: Fomin, F.V., Kratsch, D. (eds.) Exact Exponential Algorithms. TTCSAES, pp. 153–160. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16533-7_9

12. Garey, M.R., Johnson, D.S.: Computers and Intractability. A Guide to the Theory of NPcompleteness. Freeman, New York (1979)
13. Mirzaei, S., Kfoury, A.J.: Linear arrangement of Halin graphs. CoRR abs/1509.08145 (2015)
14. Tamaki, S., Yoshida, Y.: Approximation guarantees for the minimum linear arrangement problem by higher eigenvalues. In: Gupta, A., Jansen, K., Rolim, J., Servedio, R. (eds.) APPROX/RANDOM-2012. LNCS, vol. 7408, pp. 313–324. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32512-0_27