

# Recognizing a totally odd $K_4$ -subdivision, parity 2-disjoint rooted paths and a parity cycle through specified elements (extended abstract)\*

Ken-ichi Kawarabayashi<sup>†‡§</sup>

Zhentao Li<sup>¶\*\*</sup>

Bruce Reed<sup>††‡‡</sup>

## Abstract

A *totally odd  $K_4$ -subdivision* is a subdivision of  $K_4$  where each subdivided edge has odd length. The recognition of a totally odd  $K_4$ -subdivision plays an important role in both graph theory and combinatorial optimization. Sewell and Trotter [53], Zang [63] and Thomassen [60] independently conjectured the existence of a polynomial time recognition algorithm. In this paper, we give the first polynomial time algorithm for solving this problem.

We also study the *the parity two disjoint rooted paths problem* where we determine if there exists two vertex disjoint paths of a specified parity between two pairs of terminals.

Using a similar technique, we give an  $O(|E(G)||V(G)|\alpha(|E(G)|, |V(G)|))$  algorithm for the parity two disjoint rooted paths problem on an input graph  $G$ , where  $\alpha(|E(G)|, |V(G)|)$  is the inverse of the Ackermann function. We note that this clearly gives an algorithm for the well-known non-parity version of the two disjoint rooted paths problem [19, 50, 52, 55, 58].

We then extend our approach to give a polynomial time algorithm which determines, for any fixed  $k$ , whether there exists a cycle of a given parity through  $k$  independent input edges.

This generalizes the non-parity version of the algorithm in [22]. Thomassen [61] gave a polynomial algorithm for the case  $k = 2$  and hoped to use this algorithm to recognize a totally odd  $K_4$ -subdivision. Our algorithm runs in  $O(|E(G)||V(G)|\alpha(|E(G)|, |V(G)|))$  for any fixed  $k$ .

Finally, we give an  $O(|V(G)|^2 + |E(G)|\alpha(|E(G)|, |V(G)|\log|V(G)|))$  algorithm to decide whether a graph contains  $k$  disjoint paths from  $A$  to  $B$  (with  $|A| = |B| = k$ ) that are not all of the same parity.

This answers a conjecture of Thomassen [60]. This problem arises from the study of totally odd- $K_4$ -subdivisions

in 3-connected graphs [60].

**Key Words** : Parity disjoint paths, totally odd  $K_4$ -subdivision, and divide and conquer.

## 1 Introduction

In this paper, all graphs are simple and undirected. When clear from the context,  $n$  and  $m$  refer to the number of vertices and the number of edges of a graph, respectively. The function  $\alpha(m, n)$  refers to the inverse of the Ackermann function (see [54]).

A *totally odd  $K_4$ -subdivision* is a subdivision of  $K_4$  where each subdivided edge has odd length. The study of totally odd  $K_4$ -subdivisions plays an important role in both graph theory and combinatorial optimization. In graph theory, the importance of such subdivisions arises from graph coloring. Toft[62] conjectured that any graph without a totally odd  $K_4$ -subdivision is 3-colorable. This is a generalization of Hajos' conjecture for graphs without a  $K_4$ -subdivision, which was proved independently by Hadwiger [16] and Dirac [11]. Toft's conjecture received considerable attention by many researchers (for example, see [18]). It was finally settled by Zang [63] and Thomassen [61], independently. This conjecture also relates to the so-called " $t$ -perfect graphs" and "strongly  $t$ -perfect graphs" from the theory of perfect graphs. For more details, we refer the reader to Schrijver's book [48].

In combinatorial optimization, there is an interesting min-max relation in graphs with no totally odd  $K_4$ -subdivision. A graph  $G$  is said to be  $\alpha$ -critical if the cardinality  $\alpha(G)$  of its largest stable set increases with the removal of any edge. For an arbitrary graph  $G$ , denote by  $\tilde{\rho}(G)$  the minimum cost of a family of vertices, edges and odd cycles covering  $V(G)$ , where the cost of a vertex or an edge is 1, the cost of an odd cycle  $C$  is  $(|C| - 1)/2$ , and the cost of a family is the sum of the costs of its element. Clearly  $\alpha(G) \leq \tilde{\rho}(G)$ . But if  $G$  contains no totally odd  $K_4$ -subdivision, then we have  $\alpha(G) = \tilde{\rho}(G)$  as a corollary of Sewell and Trotter's theorem [53]. This answered a conjecture of Chvatál [5]. For more consequences of this result, we refer the reader to Schrijver's book [48].

Sewell and Trotter [53] gave a polynomial time

\*This work was done as a part of an INRIA-NII collaboration under MOU grant, and partially supported by MEXT Grant-in-Aid for Scientific Research on Priority Areas "New Horizons in Computing"

<sup>†</sup>National Institute of Informatics, 2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo, Japan.

<sup>‡</sup>Research partly supported by Japan Society for the Promotion of Science, Grant-in-Aid for Scientific Research, by C & C Foundation, by Kayamori Foundation and by Inoue Research Award for Young Scientists.

<sup>§</sup>Email address: k\_keniti@nii.ac.jp

<sup>¶</sup>School of Computer Science, McGill University, Montreal Canada

<sup>||</sup>Research partly supported by NSERC.

<sup>\*\*</sup>Email address: zhentao.li@mail.mcgill.ca

<sup>††</sup>Canada Research Chair in Graph Theory, McGill University, Montreal Canada and Project Mascotte, INRIA, Laboratoire I3S, CNRS, Sophia-Antipolis, France

<sup>‡‡</sup>Email address: breed@cs.mcgill.ca

algorithm to find a maximum stable set in a graph without a totally odd  $K_4$ -subdivision. Then Zang [63] gave a polynomial time algorithm to 3-color a graph without a totally odd  $K_4$ -subdivision. Since finding a 3-coloring and finding a stable set are both well-known NP-complete problems, both Sewell and Trotter [53] and Zang [63] conjectured that there is a polynomial time algorithm to recognize a totally odd  $K_4$ -subdivision (see also [60]).

In this paper, we prove this conjecture. Namely:

**THEOREM 1.1.** *There is an  $O(m\alpha(m,n)n)$  algorithm which determines if an input graph contains a totally odd- $K_4$ -subdivision and outputs a totally odd- $K_4$ -subdivision if there is one.*

We note that the problem of recognizing an odd- $K_4$ -subdivision is known to have a polynomial time algorithm [14]. An *odd- $K_4$ -subdivision* is a subdivision of a  $K_4$  where every triangle in  $K_4$  correspond to an odd cycle in the graph. Although this result was known for more than 20 years, the problem of finding a polynomial time algorithm for recognizing a totally odd- $K_4$ -subdivision beyond an odd- $K_4$ -subdivision remained open until now.

Note that we may obtain a totally odd- $K_4$ -subdivision by taking a cycle of length 4 and adding odd length paths between non-adjacent vertices of the cycle. This gives rise to the problem of determining whether such odd paths exist. We solve the following more general problem which we call the *parity 2 disjoint rooted paths (P2-DRP)*.

**Input:** A graph  $G$ , two pair of vertices  $(s_1, t_1), (s_2, t_2)$  in  $G$ , and a parities  $\ell_i \in \{0, 1\}$  for each  $i = 1, 2$ .

**Output :** If they exist, output vertex disjoint paths  $P_1, P_2$  in  $G$  where  $P_i$  joins  $s_i$  and  $t_i$  with  $|P_i| \cong \ell_i \pmod 2$  for  $i = 1, 2$ .

We prove the following.

**THEOREM 1.2.** *There is an  $O(m\alpha(m,n)n)$  algorithm for the parity two disjoint rooted paths problem.*

Note that this clearly gives an algorithm for the non-parity version of the well-known two disjoint paths problem [19, 50, 52, 55, 58]. The first algorithms for solving the undirected version of the problem were developed independently by Seymour [50], Shiloach [52] and Thomassen [58]. Their algorithms ran in time  $O(mn)$ . The running time was improved to  $O(m\alpha(m,n))$  time by Tholey [55]. Finally, Kapadia, Kawarabayashi, Kobayashi, Li and Reed [19] gave an  $O(m)$  algorithm for the problem. In view of these

results, the time complexity in Theorem 1.2 could possibly be improved.

For the non-parity case, there exists polynomial time algorithms for solving the disjoint paths problem for any fixed number of pairs of terminals [44, 41]. However, if the number of pairs of terminals is also part of the input, the problem becomes NP-complete [31].

We are currently trying to prove the parity version of this more general problem, where we are given  $k$  pairs of terminals instead of two. Recently, many researchers became interested in the parity version of the disjoint paths problem. This is since the problem relates to many deep results and new theory. E.g., the odd  $S$ -paths theorem [6, 13] (which generalizes the well-known Mader's  $S$ -paths theorem), the odd clique minor and the generalized Hadwiger's conjecture [13, 23, 25], Robertson-Seymour's Graph Minor Project and its applications to Matroid Theory (Geelen, Gerards, Whittle's project), the two disjoint odd cycles theorem by Lovász and Schrijver (see [49]), and the odd disjoint cycles problem [26, 49]. In fact, the odd disjoint cycles problem is one of the central problems in combinatorial optimization since it has connection to Matroid theory, Matching Theory, and Structure Graph Theory.

However, the parity version of the problem seems to be inherently more difficult. Unlike the non-parity case, it seems very hard to get a structure theorem for graphs without the two parity rooted paths. In the non-parity case, if an input graph  $G$  is 4-connected then the only obstruction is a plane graph with all terminals on the outer face boundary in a specific order. In the parity case, there are at least five families of graphs that may be an obstruction.

Thus our first attempt is to use the same approach as Robertson and Seymour. But the Robertson-Seymour algorithmic framework is not enough to solve this problem, since we also need to control the parity of the paths. More precisely, as with their algorithm to solve the  $k$  disjoint paths problem, in each iteration, we would like to either use a huge clique minor as a "crossbar", or exploit the structure of graphs in which we cannot find such a minor. However, we must maintain the parity of the paths and can only use an "odd clique minor". We must also describe the structure of graphs without such a minor and discuss how to exploit this structure. This motivates us to look for a huge odd clique minor to attack this problem. As we shall see, our problem needs many more techniques and ideas, beyond the Robertson-Seymour algorithmic framework.

We now turn to more general problems. Using similar approach, we shall also prove the following general results.

First, we give an algorithm for solving the following

problem.

**THEOREM 1.3.** *Let  $k$  be fixed and let  $e_1, e_2, \dots, e_k$  be  $k$  independent edges (edges not sharing an endpoint). For a given parity 0 or 1 (i.e., even or odd), there is an  $O(m\alpha(m, n) \log n + n^2)$  algorithm to decide whether or not there is a cycle  $C$  through  $k$  independent edges  $e_1, e_2, \dots, e_k$ , with the specified parity.*

This generalizes the non-parity version of the algorithm in [22]. In [61], Thomassen gave a polynomial algorithm for the case  $k = 2$ . He hoped to use this algorithm to recognize a totally odd  $K_4$ -subdivision, as his proof needs a cycle of a given parity as in Theorem 1.3 when  $k = 2$ . Our algorithm gives a polynomial time algorithm for any fixed  $k$ . Furthermore, our proof does imply that Theorems 1.3 still holds if we replace  $k$  independent edges by  $k_1$  vertices and  $k_2$  independent edges, where  $k_1 + k_2 = k$ .

Second, we prove the following result about the parity of paths in a linkage, conjectured by Thomassen [60].

**THEOREM 1.4.** *For any fixed  $k$ , there is an  $O(m\alpha(m, n) \log n + n^2)$  algorithm to decide whether or not, given two disjoint vertex sets  $A, B$  with  $|A| = |B| = k$ , a given graph contains  $k$  disjoint paths from  $A$  to  $B$ , not all of the same parity.*

In [60], Thomassen noted how this theorem also relates to the study of totally odd  $K_4$ -subdivisions. Namely, if there are two disjoint triangles in a 3-connected graph  $G$ , then either there is a totally odd  $K_4$ -subdivision in  $G$  or any three disjoint paths connecting two triangles must have the same parity in  $G$ .

Since the proofs for Theorems 1.3 and 1.4 are almost identical, we prove them simultaneously. To do so, in our proof, when we deal with Theorem 1.3, we will refer to “parity cycles” while when we deal with Theorem 1.4, we will refer to “parity disjoint paths”. In addition, if we say “a desired parity disjoint paths”, then that means that there are  $k$  disjoint paths from  $A$  to  $B$ , not all of the same parity.

## 2 Preliminaries

In this section, we state some definitions and past results that we will need. First is the notation of a separation in a graph.

A *separation*  $(A, B)$  in a graph  $G$  is two subsets of vertices  $A$  and  $B$  with  $V(G) = A \cup B$  such that there is no edge between  $A - B$  and  $B - A$ . The *order of the separation*  $(A, B)$  is  $|A \cap B|$ .

**2.1 Tree-width and Brambles** In all cases, when the input graph has bounded “tree-width” we may

use well known results to solve our problem in linear time [3, 7]. The notion of tree-width was first introduced by Halin in [15], but was unnoticed until its rediscovery by Robertson and Seymour [42] and, independently, by Arnborg and Proskurowski [1].

A *tree decomposition* of a graph  $G$  consists of a tree  $T$  and a subtree  $S_v$  of  $T$  for each vertex  $v$  of  $G$  such that if  $uv$  is an edge of  $G$  then  $S_u$  and  $S_v$  intersect. For each node  $t$  of the tree, we let  $W_t$  be the set of vertices  $v$  of  $G$  such that  $t \in S_v$ . We let  $H_t$  be the graph obtained from the subgraph of  $G$  induced by  $W_t$  by adding an edge between  $x$  and  $y$  if there is some  $s$  such that  $x, y \in W_s \cap W_t$ . The *width* of a tree decomposition is the maximum of  $|W_t|$  over the nodes  $t$  of  $T$ . The *tree-width* of a graph is the minimum of the widths over all its tree decompositions.

A *bramble*  $\beta$  is a set of trees every two of which intersect or are joined by an edge (thus a clique model (minor) is a bramble whose elements are disjoint). The *order* of a bramble  $\beta$ , denoted  $ord(\beta)$ , is the minimum size of a hitting set of its elements (that is, a set  $H$  of vertices intersecting the vertex set of each tree of  $\beta$ ). Clearly every clique model (minor) of order  $l$  is a bramble of order  $l$ . Also for any set  $W$  of vertices, the set  $\beta_W$  of trees of  $G$  containing more than half the vertices of  $W$  is a bramble since any two such trees intersect. We now characterize graphs which have no brambles of order  $l$ , using tree decompositions.

It is not hard to see that for every bramble  $\beta$  and every tree decomposition there is a node  $t$  such that  $W_t$  is a hitting set for  $\beta$ . This implies that the tree width of  $G$  is at least the maximum order of a bramble. Seymour, and Thomas showed that this bound is tight.

**THEOREM 2.1.** [51] *The maximum order of a bramble in  $G$  is equal to its tree width.*

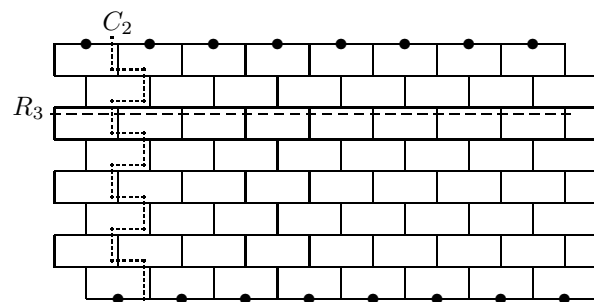


Figure 1: An elementary wall of height 8

**2.2 Wall and Grid** An elementary wall of height eight is depicted in Figure 1. An *elementary wall* of height  $h$  for  $h \geq 3$  is similar. It consists of  $h$  levels each

containing  $h$  bricks, where a brick is a cycle of length six. A wall of height  $h$  (or  $h$ -wall) is obtained from an elementary wall of height  $h$  by subdividing some of the edges, i.e. replacing the edges with internally vertex disjoint paths with the same endpoints (see Figure 2). The *nails* of a wall are the vertices of degree three within it. Any wall has a unique planar embedding. The *perimeter* of a wall  $W$ , denoted  $\text{per}(W)$  is the unique face in this embedding which contains more than 6 nails. For any wall  $W$  in a given graph  $H$ , there is a unique component  $U$  of  $H - \text{per}(W)$  containing  $W - \text{per}(W)$ . The *compass* of  $W$ , denoted  $\text{comp}(W)$ , consists of the graph with vertex set  $V(U) \cup V(\text{per}(W))$  and edge set  $E(U) \cup E(\text{per}(W)) \cup \{xy | x \in V(U), y \in V(\text{per}(W))\}$  (note that this may exclude edges between vertices of  $\text{per}(W)$ ). We note that an elementary wall of height  $h$  can be decomposed into  $k + 1$  disjoint horizontal paths, which we enumerate, from top to bottom, as  $R_1, \dots, R_{k+1}$ . It also contains  $k + 1$  columns where each column contains  $2k - 1$  edges, one from each row except the first and the last and  $k+1$  vertical edges (we omit the fussy details). We enumerate the columns from left to right as  $C_1, \dots, C_{k+1}$ . The columns and rows of arbitrary walls are defined similarly. The *corners* of a wall are  $R_1 \cap C_1, R_1 \cap C_{k+1}, R_{k+1} \cap C_1, R_{k+1} \cap C_{k+1}$ . A wall is *flat* if its compass does not contain two vertex disjoint paths connecting the diagonally opposite corners. A *subwall* of a wall  $W$  is a wall which is a subgraph of  $W$ . A subwall of  $W$  of height  $h$  is *proper* if it consists of  $h$  consecutive bricks from each of  $h$  consecutive rows of  $W$ . We say a proper subwall  $W'$  is *dividing* in a subgraph  $H$  if  $W' \subseteq H$  and the compass of  $W'$  in  $H$  is disjoint from  $(W - W') \cap H$ .

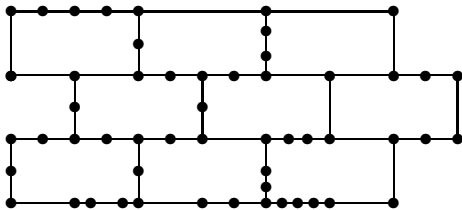


Figure 2: A wall of height 3

We now state one of the most important results about tree-width.

**THEOREM 2.2.** [43] *For any  $r$ , there exists a constant  $f(r)$  such that if  $G$  has tree-width at least  $f(r)$  (equivalently a bramble  $\beta$  of order at least  $f(r)$ ), then  $G$  contains a wall  $W$  of height  $r$ .*

The best known upper bound for  $f(r)$  is  $2^{205r}$  [10,

39, 47] and the best known lower bound is  $\Theta(r^2 \log r)$  [47].

**THEOREM 2.3.** *Given a graph  $G$  with tree-width at least  $f_1(r)$ , we can find a wall  $W$  of height  $r$  in linear time.*

We now outline this algorithm. By the algorithm in [37], in linear time, we can find a subgraph  $G'$  of  $G$  of tree-width at least  $f_1(r)$  and a tree decomposition of  $G'$  of width at most  $2f_1(r)$ . Then, since  $G'$  has a wall of height  $r$ , it can be found in linear time by the dynamic programming method [4].

**2.3 Odd minors** We've mentioned in the introduction that we need the notion of an "odd minor" for all our algorithms.

Recall that a graph  $H$  is a *minor* of  $G$  if  $H$  can be obtained from  $G$  using edge contraction and vertex deletion. Equivalently,  $H$  is a minor of  $G$  precisely if there are  $|V(H)|$  vertex-disjoint trees in  $G$ , one tree  $T_v$  for each vertex  $v$  of  $H$ , such that for every edge  $e = \{v, w\}$  in  $H$  there is an edge  $\hat{e}$  in  $G$  connecting the two corresponding trees  $T_v$  and  $T_w$ . Now  $H$  is an *odd minor* of  $G$  if, in addition, all the vertices of the trees can be two-colored in such a way that (1) the edges of each tree  $T_v$  are bichromatic, while (2) the edge  $\hat{e}$  connecting trees  $T_v$  and  $T_w$  corresponding to each edge  $e = \{v, w\}$  of  $H$  is monochromatic. In particular, the class of odd- $H$ -minor-free graphs (excluding a fixed graph  $H$  as an odd minor) contains the class of  $H$ -minor-free graphs (excluding a fixed graph  $H$  as a minor).

In fact, this containment is strict. The complete bipartite graph  $K_{n/2, n/2}$  certainly contains a  $K_k$ -minor for  $k \leq n/2$ , but on the other hand, it does not contain  $K_k$  as an odd minor for any  $k \geq 3$ . Also, any  $K_k$ -minor-free graph  $G$  is  $O(k\sqrt{\log k})$ -degenerate [56, 57, 33, 34], i.e. every induced subgraph has a vertex of degree at most  $O(k\sqrt{\log k})$ . Thus, any  $K_k$ -minor-free graph  $G$  has  $O(k\sqrt{\log kn})$  edges. On the other hand, some odd- $K_k$ -minor-free graphs such as  $K_{n/2, n/2}$  may have  $\Theta(n^2)$  edges.

### 3 Overview of the proof

For all our algorithms, we need to be able to find an odd minor (or some structure about the input graph). It turns out that we have a nice structure theorem which tells us that if a graph contains a huge clique minor, then either we can find a big odd clique minor or a set  $X$  of vertices of bounded size such that the component of  $G - X$  containing most of the nodes of the huge clique minor is "essentially" bipartite. This is proved using a recent result by Geelen et al. [13] (Actually, Theorem 6.1 essentially follows from the result in [13].).

In summary, roughly speaking, we have the following.

**MAIN STRUCTURAL RESULT .** *Given an input graph  $G$  we can find either*

1. *A tree decomposition of  $G$  of bounded width,*
2. *a huge clique minor and a huge odd clique minor,*
3. *a huge clique minor, but no huge odd clique minors, or*
4. *a flat wall of huge height, but no huge clique minor in  $O(m\alpha(m, n))$*

To obtain one of these structures in  $G$ , we first apply Bodlaender's algorithm to determine the treewidth of  $G$  and construct a tree decomposition if the width is small [3]. If the treewidth is large, we apply the algorithm described in Section 4 in order to determine if  $G$  contains a large clique minor. If not, the algorithm outputs a wall of large height. But if  $G$  contains a large clique minor then we further distinguish between the cases 2 and 3 by applying the algorithm in Section 6 which uses the algorithm in Section 5. If an odd clique minor is found, we output it. Otherwise, the algorithm output a structure for  $G$  which shows that  $G$  is "nearly bipartite". In this case, we make use of this structure to solve the problem under consideration.

In the remainder of this section, we assume the Main Structural Result and give an overview of each individual algorithm.

**3.1 Parity two disjoint rooted paths** In most cases, our algorithm will actually find a more restrictive set of paths which we call *variable length paths*.

**DEFINITION 3.1.** *A variable length path from  $x$  to  $y$  is a path from  $x$  to  $y$  together with an odd cycle  $C$  which meets the path in at least two vertices.*

Clearly if vertex disjoint variable length paths exist between  $(s_1, t_1)$  and  $(s_2, t_2)$  then vertex disjoint paths of any parity exist between them (by choosing to route the paths through either side of the odd cycle).

If the input graph  $G$  has low treewidth, we can easily solve the problem in linear time by using dynamic programming or by determining the satisfiability of a LinEMSOL formula (a variant of MSO2 formula, see [7]).

Otherwise, if  $G$  has large treewidth, we use a recursive algorithm which finds either

1. vertex disjoint variable length paths between each pair  $(s_1, t_1), (s_2, t_2)$ ,

2. an irrelevant vertex  $v \in V(G)$  (that is, a vertex whose removal does not change the solution to the problem), or
3. a minimal cutset  $X$  of size at most 3 where the union of components  $U$  of  $G - X$  which do not contain any vertex of  $A$  has at least 2 vertices.

In the first case, we can simply return the paths of the correct parity. In the second case, we recurse on  $G - v$ . In the third case, we replace  $U$  by a set of edges and  $P_3$ 's between the vertices of  $X$  depending on the parity of the paths which exist between the vertices of  $X$  in  $U$ .

To find one of these three outputs, we apply the Main Structural Result.

If  $G$  has a large odd clique minor then we simply use some of the vertex images of the minor as two disjoint odd cycles and others as a "crossbar" to get variable length paths. Note that by definition, any three vertex images of an odd clique minor gives an odd cycle. Thus, if we are able to find the paths from the terminals to the large odd minor, we found variable length paths. If not, we can find an irrelevant vertex inside the large odd minor.

If  $G$  has a large clique minor but no large odd clique minor then we apply Theorem 6.3 which states that such graphs are nearly bipartite (up to 1-cuts after removing a finite set of vertices). We then attempt to route paths between  $s_i$  and  $t_i$  through the large clique minor. We further require that these paths go through a block containing an odd cycle. If such paths exist then we have found variable length paths. If not, we can find a small cutset separating  $A$  and the blocks from the large clique minor. In this case, we can find an irrelevant vertex.

If  $G$  contains a large flat wall then we either find variable length paths or an irrelevant vertex away from the perimeter of the flat wall. This is the case that will require the most work.

**3.2 Totally odd- $K_4$ -subdivision** Again, we apply the Main Structural Result.

If the input graph  $G$  has low treewidth, we can easily solve the problem in linear time by using dynamic programming or by determining the satisfiability of a LinEMSOL formula.

If  $G$  has a large odd clique minor then we simply use some of the vertex images of the minor as odd cycle and others as vertex images of  $K_4$ . This allows us to find 6 variable lengths paths between our vertex images.

If  $G$  has a large clique minor but no large odd clique minor, then we apply Theorem 6.3 which states that such graphs are nearly bipartite (up to 1-cuts after

removing a finite set of vertices). We then attempt to route paths between 6-tuples of blocks and the clique minor. If we succeed, we again use these cycles to form 6 variable length paths for the totally odd- $K_4$ -subdivision and use other vertex images of the large clique minor for vertex images of the  $K_4$ . If we fail, we find a small cutset separating the odd cycles from the large clique minor. This allows us to find an irrelevant vertex inside the large clique minor.

If  $G$  contains a large flat wall, we look for an irrelevant vertex away from the perimeter of the flat wall and show that we can find a totally odd- $K_4$ -subdivision when such vertex does not exist. Again, this is the case that will require the most work.

### 3.3 Parity cycle through specified elements and paths of different parity between sets of vertices

As we mentioned in section 1, we shall prove Theorems 1.3 and 1.4 simultaneously. We sketch these algorithms in this section.

We extend the approach used by Reed [38] for building a tree decomposition of width at most  $4k$  in a graph of tree width  $k$ .

We construct a special tree-decomposition, but not necessarily the one for graphs of bounded tree-width. Each bag  $W_t$  of the decomposition is required to have be either

- a graph of bounded size, or
- a “nearly” bipartite graph (i.e, there is a vertex set  $X$  of order at most  $f(k)$  (for some function of  $k$ , to be determined later) whose removal yields a bipartite piece).

Our tree decomposition still requires adjacent nodes to intersect in a bounded number of vertices.

To build our special tree decomposition, we start by putting all vertices of the graph  $G$  in a single root node. We then iteratively divide the leaf nodes in half. However, in some cases, we apply a reduction. That is, we locally replace a subgraph by a smaller subgraph, throwing away many (at least half) vertices of the original graph. The reduction that we apply will depend on the structure of  $W_t$ .

If a leaf has bounded size or is “nearly” bipartite, we mark it so that we no longer consider it in the future.

We complete each iteration in  $O(m\alpha(m, n))$ . Thus we can construct the tree-decomposition in  $O(m\alpha(m, n) \log n)$  time, because at each iteration the leaves of the decomposition are halved (or no longer need to be considered) so that after  $2 \log n$  iterations, all the leaves have bounded size.

We then use dynamic programming to solve our problems using the tree-decomposition in  $O(n^2)$  time.

We now describe the algorithm which either divides a leaf  $W_t$  or finds a reduction inside  $W_t$ . We abuse notation and write  $W_t$  to also mean the graph induced by the vertices in  $W_t$ . We note that our algorithm never considers the internal nodes of the tree-decomposition. This is one of the important points. We only consider the leaves of the tree-decomposition.

We try to split  $W_t$  so that each child has at most half of the vertices of  $W_t$ . If we are successful, we repeat this process in the next iteration. The only time we fail is when the tree-width of  $W_t$  is too large. In that case,  $W_t$  contains a wall of huge height [10, 39, 43, 47]. Using this wall, we either get a nearly bipartite graph in  $W_t$ , which allows us to split  $W_t$  or stop (because the leaf is already in the final desired form), or we are able to make a reduction where we delete half the vertices of  $W_t$ . To make a reduction, we need a wall in  $W_t$  which is “attached” to a specific highly connected part of the graph  $W_t$ . This is done using Reed’s result[38] together with a related algorithmic result, [28].

We now sketch the algorithm used when  $W_t$  has large tree-width.

We either make a reduction to throw away half of the vertices of  $W_t$ , or notice that we can stop splitting  $W_t$ . To do so, we apply the Main Structural Result to  $W_t$ .

In all cases, the structure we obtain from the Main Structural Result (e.g., a minor or a wall) is “attached” to a specific highly connected part of  $W_t$ . This is what allows us to throw away almost half of the vertices of  $W_t$  in the case of a reduction.

Suppose  $W_t$  contains a huge odd clique minor. We show that either there are a desired parity cycle and desired parity disjoint paths, or there is a reduction we can make using the odd clique minor to throw away half of the vertices of  $W_t$ . In case we make a reduction, we use the fact that the huge odd clique minor is “attached” to a specific highly connected part of the graph  $W_t$ .

Suppose  $W_t$  contains a huge clique minor, but no huge odd clique minor. In this case, we can conclude that a graph  $N$  in  $W_t$ , which is “attached” to a specific highly connected part of the graph  $W_t$  is “essentially” bipartite.  $N$  becomes an internal node  $t$  in the tree-decomposition. Since  $N$  is attached to a specific highly connected part of the graph  $W_t$ , all the non-bipartite subgraphs in  $W_t$  are cut off by small separation from  $N$ , and moreover, each of these subgraphs contains at most half of the vertices of  $W_t$ . Thus each of these components becomes a child of  $t$ .

Finally, suppose  $W_t$  contains a huge flat wall  $W$ , which is “attached” to a specific highly connected part of the graph  $W_t$ . In this case, we can find such a wall in linear time[29]. In this case, we further distinguish

between the following two cases.

- (a)  $W_t$  contains a flat wall of huge height with many disjoint odd faces, or
- (b)  $W_t$  contains a bipartite flat wall of huge height, with many disjoint “parity breaking” paths.

In case (a),  $W_t$  has many disjoint odd faces so either a desired parity cycle and desired parity disjoint paths exist, or we can make a reduction, throwing away half of the vertices of  $W_t$ , as we did for the huge odd clique minor case.

In case (b),  $W_t$  contains a huge flat proper subwall  $W'$  of  $W$ , which is bipartite. In this case, by [13], either the graph “attached” to a specific highly connected part of  $W_t$  is “essentially” bipartite in  $W_t$  in which case we stop splitting  $W_t$ , or there are many parity breaking paths, each joining two points on the perimeter of  $W'$  in  $W_t$ . By a *parity breaking path*, we mean a path  $P$  with endpoints on the perimeter of the wall  $W'$  such that the path  $P$  together with the wall  $W'$  yields an odd cycle. In the first case, this allows us to continue to construct the tree-decomposition, as we did when  $W_t$  contains a huge odd clique minor. In the second case, either a desired parity cycle and desired parity disjoint paths exist, or we can make a reduction, throwing away half of the vertices of  $W_t$ , as we did for the huge odd clique minor case.

With some technical details found in [30], we can apply reductions to all leaves simultaneously. This allows us to construct a tree-decomposition such that each piece is either nearly bipartite or of bounded size, in  $O(m\alpha(m, n) \log n)$  time.

In order to find an odd clique minor or parity breaking paths, we need to find an augmenting path  $f(k)$  times for some function  $f$  of  $k$ . As we show in Section 5, finding an augmenting path takes  $O(m\alpha(m, n))$ . Thus, since we need at most  $2 \log n$  iterations, the total running time building the special tree decomposition is  $O(m\alpha(m, n) \log n)$ .

We now describe the algorithm which makes use of the special tree decomposition we built in  $O(n^2)$  time.

This algorithm recursively solves the “all pairs parity disjoint rooted path” problem. That is, given a graph  $G$  and a set  $T$  of terminals, for all partitions of  $T$  into sets of size two  $(s_1, t_1), \dots, (s_k, t_k)$  and all parities  $\ell_1, \dots, \ell_k$ , we wish to know if there are vertex disjoint paths of parity  $\ell_i$  between  $s_i$  and  $t_i$ .

The terminals change depending on the node of our special tree decomposition under consideration. However, in all cases, the number of terminals is bounded (by some global bound to be determined later).

Our ultimate goal is to obtain the solution to all pairs parity disjoint rooted path where the graph is the input graph to our original problem and

- $T$  is  $A \cup B$  if the original problem consists of determining if vertex disjoint paths, all of the same parity, exist between  $A$  and  $B$ , or
- $T$  is the endpoints of the edges  $e_1, \dots, e_k$  if we were asked for a cycle of specified parity through  $e_1, \dots, e_k$ .

We now describe how we obtain this solution. Let  $Y$  be either  $A \cup B$  or the endpoints of  $e_1, \dots, e_k$ , depending on the original problem.

Recall that for each node  $t$  of the special tree decomposition, either

1.  $|W_t| \leq f(k)$ , or
2.  $W_t$  is nearly bipartite. More precisely,  $W_t - X$  is bipartite for a vertex set  $X$  of order at most  $f(k)$ .

Also,  $W_t \cap W_s \subseteq X$ , where  $s$  is the parent of  $t$  in the tree.

We now work our way up from the leaves of the tree-decomposition. To solve the problem at  $t$  given the solution for its children, we let  $T = (W_s \cap W_t) \cup (Y \cap W_t)$  where  $s$  is the parent of  $t$ . In many cases, we can replace the children by a graph of bounded size which gives the same solution to the rooted problem.

If  $W_t$  has bounded size, we use the standard dynamic programming algorithm [4].

If  $W_t$  is “nearly” bipartite, we solve the disjoint paths problem using known algorithms [29, 44] to replace the children of  $t$  by graphs of bounded size. Note that we only need the non-parity version of the disjoint paths problem for nearly bipartite graphs since  $|X|$  is bounded and in the bipartite subgraph  $W_t - X$ , we do not have to worry about the parity of the paths. Moreover, we only need to consider the case where all the terminals are in  $X$ , and hence the number of terminals is also bounded. Let us observe that at the moment, the best known algorithm for the disjoint paths problem is due to [29], which runs in  $O(n^2)$  time.

Thus, we are able to solve the given problems using dynamic programming on our special tree decomposition in  $O(n^2)$ .

A more detailed description of our algorithm is provided in [30].

#### 4 Finding a large flat wall in $H$ -minor-free graphs

In this section and the next two, we give an overview of the proof of the Main Structural Result. First, we make

use of a structural result of Robertson and Seymour for graph with large treewidth with no large clique minor. But to state their result, we need some definitions.

**DEFINITION 4.1.** *We say that  $G$  can be embedded into a plane, up to 3-separations, if, for some  $k \geq 0$ , there is a partition of  $V(G)$  into  $C, A_1, \dots, A_k$  such that*

- (1) *for  $1 \leq i < j \leq k$ , there is no edge between a vertex of  $A_i$  and a vertex of  $A_j$ ,*
- (2) *for  $1 \leq i \leq k$ ,  $|N(A_i)| \leq 3$ ,*
- (3)  *$|C| > 0$ , and*
- (4) *removing each  $A_i$  and adding all missing edges between the vertices of  $N(A_i)$  yields a graph  $G'$  which is embeddable in the plane.*

Note that if  $G$  can be embedded in the plane up to 3-separations then we can choose  $A_1, \dots, A_k$  so that whenever  $|A_i| = 3$ ,  $N(A_i)$  is a facial triangle in  $G'$ . Indeed, if  $N(A_i)$  is not a facial triangle then  $N(A_i)$  is a 3-separation in  $G'$  and we can add all but one component of  $G' - N(A_i)$  to the list as  $A_{k+1}, A_{k+2}, \dots$  (but actually, there are at most two components since  $G'$  is planar).

Also note that  $G'$  may not be a minor of  $G$  since a vertex of degree 3 can be replaced by the triangle (the so called  $Y - \Delta$  interchange). But in fact, this is the only reason  $G'$  would not be a minor of  $G$ . That is if we do not allow  $|N(A_i)| = 3$  and  $|A_i| = 1$  for any  $i$  then  $G'$  is a minor of  $G$ .

Note that if the compass of  $W$  has a planar embedding whose infinite face is bounded by the perimeter of  $W$  then  $W$  is clearly flat. Seymour [50], Thomassen [58], and others have characterized precisely which walls are flat.

**THEOREM 4.1.** [50, 58] *If a wall  $W$  is flat then its compass can be embedded in a plane, up to 3-separations, so that its perimeter is the outer face boundary.*

Finally, we are ready to state the main results of this section.

**THEOREM 4.2.** [44] *For every pair of integers  $l$  and  $t$  there exist integers  $w(l, t) > w'(l, t) > \max(l, t)$  such that the following holds. If the tree-width of a graph  $G$  is at least  $w(l, t)$ , and  $G$  has no clique minor of order  $t$ , then there is a wall  $H$  of height  $w'(l, t)$ , and for some subset  $X$  of less than  $\binom{t}{2}$  vertices of  $G$  there is a proper subwall  $W$  (of the wall  $H$ ) of height  $l$ , which is disjoint from  $X$  and is flat and dividing in  $G - X$ .*

The algorithmic version of this theorem is as follows.

**THEOREM 4.3.** [44] *For any  $t \geq 0$  and  $h \geq 2$ , there is a computable constant  $f_2(t, h)$  such that the following can be done in  $O(mn)$  time, where  $m$  is the number of edges of a give graph  $G$ .*

*Input:* A graph  $G$ , a wall  $H$  of height at least  $f_2(t, h)$ .

*Output:* Either

1. *a  $K_t$ -minor, or*
2.
  - *a subset  $X \subseteq V(G)$  of order at most  $\binom{t}{2}$ ,*
  - *$t^2$  proper subwalls  $H_1, \dots, H_{t^2}$  of height  $h$  which are flat and dividing in  $G - X$ , and*
  - *a flat embedding of  $H_i$  (in  $G - X$ ) for each  $i$ .*

The running time of the algorithm in Theorem 4.3 has been improved to  $O(m)$  [29].

We can combine Theorem 4.3 and Theorem 2.3 to obtain the following.

**COROLLARY 4.1.** *For any  $t \geq 0$  and  $h \geq 2$ , there is a computable constant  $f_3(t, h)$  such that the following can be done in  $O(m)$  time, where  $m$  is the number of edges of a give graph  $G$ .*

*Input:* A graph  $G$ .

*Output:* Either

1.  *$G$  has tree-width at most  $f_3(t, h)$ , or*
2. *a  $K_t$ -minor, or*
3.
  - *a subset  $X \subseteq V(G)$  of order at most  $\binom{t}{2}$ ,*
  - *$t^2$  proper subwalls  $H_1, \dots, H_{t^2}$  of height  $h$  which are flat and dividing in  $G - X$ , and*
  - *a flat embedding in  $H_i$  (in  $G - X$ ) for each  $i$ .*

## 5 Odd $S$ -paths

We make use the following theorem both in the proof of the Main Structural Result and the algorithms described in Section 3.

**THEOREM 5.1.** [13] *For any set  $S$  of vertices of a graph  $G$ , either*

1. *there are  $k$  disjoint odd  $S$  paths, i.e.,  $k$  disjoint paths each of which has an odd number of edges and both its endpoints in  $S$ , or*
2. *there is a vertex set  $X$  of order at most  $2k - 2$  such that  $G - X$  contains no such paths.*

The proof given in [13] actually gives a polynomial time algorithm for finding either the paths or vertex set in Theorem 5.1.

This algorithm runs in time  $O(km\alpha(m, n))$  since finding an augmenting path takes  $O(m\alpha(m, n))$  and we

must do this  $k$  times (or fail at some point and find a vertex set  $X$ ).

In our algorithms,  $k$  is always fixed so we can obtain one of the two outputs in nearly linear time.

## 6 Obtaining a huge odd clique minor

In this section, we use Theorem 5.1 to obtain a large odd clique minor given a clique minor of order  $\ell$ , controlled by the bramble  $\beta_G$  where  $\ell \geq 32k\sqrt{\log k}$ .

We say that a  $K_\ell$ -model in  $G$  is *even* if the image of  $K_\ell$  in  $G$  is bipartite. In this case, we say that  $G$  contains an even  $K_\ell$ -minor.

We can now state the main result of this section.

**THEOREM 6.1.** *Suppose  $G$  has a  $K_{32k\sqrt{\log k}}$ -minor. Then either  $G$  has an odd complete minor of order  $k$  or  $G$  has a vertex set  $X$  with  $|X| < 8k$  such that the (unique) block  $F$  intersecting all but at most  $8k$  disjoint trees of the even clique minor in  $G - X$  is an induced bipartite graph, and each odd cycle in  $G - X$  is contained in either components of  $G - X$  that do not intersect  $F$  or blocks with a cut vertex to the block  $F$ .*

This theorem differs from the one in [13] only in the hypothesis. In [13], an even complete minor of order  $16k$  is required.

**THEOREM 6.2.** [13] *Suppose  $G$  has an even complete minor of order  $16k$ . Then either  $G$  has an odd complete minor of order  $k$  or  $G$  has a vertex set  $X$  with  $|X| < 8k$  such that the (unique) block  $F$  intersecting all but at most  $8k$  disjoint trees of the even clique minor in  $G - X$  induces a bipartite graph, and each odd cycle in  $G - X$  is contained in either components of  $G - X$  that do not intersect  $F$  or blocks with a cut vertex to the block  $F$ .*

A simple combinatorial argument shows that a sufficiently large complete minor contains a  $K_k$  even minor, thus the assumption of Theorem 6.2, which requires a large  $K_k$  even minor, can be changed to assuming a sufficiently large complete minor. The detailed proof is provided in [30].

If  $G$  contains a  $K_{32k\sqrt{\log k}}$ -minor but no odd  $K_{16k}$ -minor then there are at least  $18k$  nodes of the  $K_{32k\sqrt{\log k}}$ -minor that are contained in  $F$  since  $F$  intersects all but  $8k$  disjoint trees of the even  $K_{16k}$ -minor in  $G - X$  and  $|X| \leq 8k$ ,

Therefore we obtain the following which brings us one step closer to our the Main Structural Result.

**THEOREM 6.3.** *Suppose  $G$  contains a  $K_{32k\sqrt{\log k}}$ -minor then either*

1.  $G$  has an odd- $K_k$ -minor, or

2.  $G$  has a vertex set  $X$  of order at most  $8k$  such that the (unique) component  $R$  of  $G - X$  containing an element of the bramble  $\beta_G$  contains all but at most  $8k$  nodes of the  $K_{32k\sqrt{\log k}}$ . Moreover, the component  $R$  can be written as  $F \cup B_1, \dots$  such that the following holds:

- (a)  $F$  is an induced bipartite graph and contains at least at least  $18k$  nodes of the  $K_{32k\sqrt{\log k}}$ -minor, and
- (b) each  $B_i$  contains an odd cycle.

Furthermore, we can find one of these structures in  $O(m\alpha(m, n))$ .

## 7 Conclusions

We are now ready to present the the Main Structural Result. We just need to combine Theorem 6.3 and Corollary 4.1, where  $t = 32k\sqrt{\log k}$  in Corollary 4.1.

**THEOREM 7.1.** *For any fixed  $k$  and  $h \geq 2$ , there is a computable constant  $f_4(k, h)$  Corollary such that the following can be done in  $O(m\alpha(m, n))$ .*

*Input:* A graph  $G$ .

*Output:* Either

1.  $G$  has tree-width at most  $f_4(k, h)$ , or
2.  $G$  has an odd- $K_k$ -minor, or
3.  $G$  has a vertex set  $X$  of order at most  $8k$  such that the (unique) component  $R$  of  $G - X$  containing an element of the bramble  $\beta_G$  contains all but at most  $8k$  nodes of the  $K_{32k\sqrt{\log k}}$ . Moreover, the component  $R$  can be written as  $F \cup B_1, \dots$  such that
  - (a)  $F$  is an induced bipartite graph and contains at least at least  $18k$  nodes of the  $K_{32k\sqrt{\log k}}$ -minor, and
  - (b) each  $B_i$  contains an odd cycle, or
4.
  - a subset  $X \subseteq V(G)$  of order at most  $\binom{32k\sqrt{\log k}}{2}$ ,
  - $500k^2 \log k$  proper subwalls  $H_1, \dots, H_{500k^2 \log k}$  of height  $h$  which are flat and dividing in  $G - X$ , and
  - a flat embedding in  $H_i$  (in  $G - X$ ) for each  $i$ .

Here  $f_4(k, h) = f_3(32k\sqrt{\log k}, h)$  where  $f_3$  is the same as in Corollary 4.1.

As we overviewed in Section 3, we use this theorem to prove Theorems 1.1, 1.2, 1.3 and 1.4.

We are currently trying extend our algorithm to solve the parity version of the disjoint paths problem

for general  $k$ . This would generalize Robertson and Seymour's groundbreaking work. Recently, many researchers became interested in the parity version of the disjoint paths problem. This is since the problem relates to many deep results and new theory.

It seems us that Theorem 7.1 helps a lot, but we still need to extend all the arguments in graph minor theory to a parity version. This would need a lot of space and time.

## References

- [1] S. Arnborg and A. Proskurowski, Linear time algorithms for NP-hard problems restricted to partial  $k$ -trees, *Discrete Appl. Math.* **23** (1989), 11–24.
- [2] T. Böhme, K. Kawarabayashi, J. Maharry and B. Mohar, Linear connectivity forces large complete bipartite minors, *J. Combin. Theory Ser. B.* **99** (2009), 557–582.
- [3] H. L. Bodlaender, A linear-time algorithm for finding tree-decomposition of small treewidth, *SIAM J. Comput.* **25** (1996), 1305–1317.
- [4] H. L. Bodlaender, Dynamic Programming Algorithms on Graphs of Bounded Tree-Width, In *Proceedings of the 15th International Colloquium on Automata, Languages and Programming*, volume 317 (1988), 105–118.
- [5] V. Chvátal, On certain polytopes associated with graphs, *J. Combin. Theory Ser. B* **18** (1975), 138–154.
- [6] M. Chudnovsky, J. Geelen, B. Gerards, L. Goddyn, M. Lohman and P. Seymour, Packing non-zero  $A$ -paths in group labelled graphs, *Combinatorica* **26** (2006), 521–532.
- [7] B. Courcelle and J. A. Makowsky and U. Rotics, Linear time solvable optimization problems on graphs of bounded clique-width, *Theory of Computing Systems* **33** (2000), 125–150.
- [8] I. Dejter and V. Neumann-lara, Unboundedness for generalized odd cycle traversability and a Gallai conjecture, paper presented at the Fourth Caribbean Conference on Computing, Puerto Rico, 1985.
- [9] E. D. Demaine, M. Hajiaghayi, and K. Kawarabayashi, Algorithmic graph minor theory: Decomposition, approximation and coloring, Proc. 46th Ann. IEEE Symp. Found. Comp. Sci., Pittsburgh, PA, 2005, pp. 637–646.
- [10] R. Diestel, K. Yu. Gorbunov, T. R. Jensen, and C. Thomassen, Highly connected sets and the excluded grid theorem, *J. Combin. Theory Ser. B* **75** (1999), 61–73.
- [11] G. A. Dirac, A property of 4-chromatic graphs and some remarks on critical graphs, *J. London Math. Soc.*, **27** (1952), 85–92.
- [12] P. Erdős and L. Pósa, On the maximal number of disjoint circuits of a graph, *Publ. Math. Debrecen* **9** (1962), 3–12.
- [13] J. Geelen, B. Gerards, B. Reed, P. Seymour and A. Vetta, On the odd variant of Hadwiger's conjecture, *J. Combin. Theory Ser. B* **99** (2009), 20–29
- [14] B. Gerards, An extension of Konig's theorem to graphs with no odd- $K_4$ . *J. of Combin. Theory, Ser. B* **47** (1989), 330–348.
- [15] R. Halin,  $S$ -function for graphs, *J. Geometry* **8** (1976), 171–186.
- [16] H. Hadwiger, Über eine Klassifikation der Streckenkomplexe, *Vierteljahrsschr. Naturforsch. Ges. Zürich*, **88** (1943), 133–142.
- [17] T. Ibaraki and H. Nagamochi, A linear time algorithm for finding a sparse  $k$ -connected spanning subgraph of a  $k$ -connected graph, *Algorithmica* **7**, (1992), 583–596.
- [18] T. R. Jensen and F.B. Shepherd, Note on a conjecture of Toft, *Combinatorica* **15** (1995), 373–377.
- [19] R. Kapadia, K. Kawarabayashi, Y. Kobayashi, Z. Li and B. Reed, Faster algorithms for the disjoint paths problem *submitted*.
- [20] K. Kawarabayashi, On the connectivity of minimal counterexamples to Hadwiger's conjecture, *J. Combin. Theory Ser. B.* **97** (2007), 144–150.
- [21] K. Kawarabayashi, Rooted minors problem in highly connected graphs, *Discrete Math* **287** (2004) 121–123.
- [22] K. Kawarabayashi, Improved algorithm for find a cycle through elements, *IPCO'08*, 374–384.
- [23] K. Kawarabayashi, Note on coloring graphs with no odd- $K_k$ -minors, *J. Combin. Theory Ser. B.* **99** (2009), 738–741.
- [24] K. Kawarabayashi and B. Reed, Highly parity linked graphs, *Combinatorica* **29** (2009), 215–225.
- [25] K. Kawarabayashi and Z. Song, Some remarks on the odd Hadwiger's conjecture, *Combinatorica.* **27** (2007), 429–438.
- [26] K. Kawarabayashi and P. Wollan, Non-zero disjoint cycles in highly connected group-labelled graphs, *J. Combin. Theory Ser. B.* **96**, (2006), 296–301.
- [27] K. Kawarabayashi and A. Nakamoto, The Erdős-Pósa property for odd cycles on an orientable fixed surface, *Discrete Math.* **307** (2007), 764–768.
- [28] K. Kawarabayashi and B. Reed, A nearly linear time algorithm for the half disjoint paths packing, *ACM-SIAM Symposium on Discrete Algorithms (SODA'08)*, 446–454.
- [29] K. Kawarabayashi, Y. Kobayashi and B. Reed, The disjoint paths problem in quadratic time, *submitted*.
- [30] K. Kawarabayashi, Z. Li and B. Reed, Recognizing a totally odd  $K_4$ -subdivision, parity 2-disjoint rooted paths and a parity cycle through specified elements (journal version) *in preparation (draft available at url)*
- [31] R. M. Karp, On the computational complexity of combinatorial problems, *Network*, **5**, (1975), 45–68.
- [32] J. Kleinberg, Decision algorithms for unsplittable flows and the half-disjoint paths problem, Proc. 30th ACM Symposium on Theory of Computing, 1998. 530–539.
- [33] A. Kostochka, Lower bound of the Hadwiger number of graphs by their average degree, *Combinatorica* **4** (1984), 307–316.

- [34] A. Kostochka, The minimum Hadwiger number for graphs with a given mean degree of vertices (in Russian), *Metody Diskret. Analiz.* **38** (1982), 37–58.
- [35] M. Middendorf and F. Pfeiffer, On the complexity of the disjoint paths problem, *Combinatorica*, **13** (1993), 97–107.
- [36] B. Mohar and C. Thomassen, *Graphs on Surfaces*, Johns Hopkins University Press, Baltimore, MD, 2001.
- [37] L. Perkovic and B. Reed, An improved algorithm for finding tree decompositions of small width; *International Journal on the Foundations of Computing Science.*, **11**, (2000), 81–85.
- [38] B. Reed, Finding approximate separators and computing tree width quickly; STOC 1992, Victoria B.C., 1992.
- [39] B. Reed, Tree width and tangles: a new connectivity measure and some applications, in “*Surveys in Combinatorics, 1997 (London)*”, London Math. Soc. Lecture Note Ser. **241**, Cambridge Univ. Press, Cambridge, 1997, pp. 87–162.
- [40] B. Reed, Mangoes and blueberries, *Combinatorica* **19** (1999), 267–296.
- [41] B. Reed, Rooted Routing via Graph Minors, to appear.
- [42] N. Robertson and P. D. Seymour, Graph minors. II. Algorithmic aspects of tree-width, *J. Algorithm* **7** (1986), 309–322.
- [43] N. Robertson and P. D. Seymour, Graph minors. V. Excluding a planar graph, *J. Combin. Theory Ser. B* **41** (1986), 92–114.
- [44] N. Robertson and P. D. Seymour, Graph minors. XIII. The disjoint paths problem, *J. Combin. Theory Ser. B* **63** (1995), 65–110.
- [45] N. Robertson and P. D. Seymour, Graph minors. XVI. Excluding a non-planar graph, *J. Combin. Theory Ser. B* **89** (2003), 43–76.
- [46] N. Robertson and P. D. Seymour, An outline of a disjoint paths algorithm, in: “*Paths, Flows, and VLSI-Layout*,” B. Korte, L. Lovász, H. J. Prömel, and A. Schrijver (Eds.), Springer-Verlag, Berlin, 1990, pp. 267–292.
- [47] N. Robertson, P. D. Seymour and R. Thomas, Quickly excluding a planar graph, *J. Combin. Theory Ser. B* **62** (1994), 323–348.
- [48] A. Schrijver, *Combinatorial Optimization: Polyhedra and Efficiency*, number 24 in Algorithm and Combinatorics, Springer Verlag, 2003.
- [49] P. Seymour, Matroid minors, *Handbook of Combinatorics*, (Eds.: R. L. Graham, M. Grötschel and L. Lovász). North-Holland, Amsterdam, 1985, 419–431.
- [50] P. Seymour, *Disjoint paths in Graphs*, *Discrete Mathematics* **29** (1980), 293 - 309.
- [51] P. Seymour and R. Thomas, Graph searching and a min-max theorem for tree-width, *J. Combin. Theory Ser. B* **58** (1993), 22–33.
- [52] Y. Shiloach, A polynomial solution to the undirected two paths problem, *J. ACM*, **27** (1980), 445–456.
- [53] E.C. Swell and L. E. Trotter, Stability critical graphs and even subdivision of  $K_4$ , *J. Combin. Theory Ser. B* **59** (1993), 74–84.
- [54] R.E. Tarjan, *Data Structures and network algorithms*, SIAM, Philadelphia, PA, 1983.
- [55] T. Tholey, Solving the 2-disjoint paths problem in nearly linear time, *Theory of computing systems* **39** (2004), 51–78.
- [56] A. Thomason, An extremal function for contractions of graphs, *Math. Proc. Cambridge Philos. Soc.* **95** (1984), 261–265.
- [57] A. Thomason, The extremal function for complete minors, *J. Combin. Theory Ser. B* **81** (2001), 318–338.
- [58] C. Thomassen, *2-linked graph*, *European Journal of Combinatorics* **1** (1980), 371 - 378.
- [59] C. Thomassen, *On the presence of disjoint subgraphs of a specified type*, *Journal of Graph Theory*, **12** (1988), 101 - 111.
- [60] C. Thomassen, Parity, cycle space, and  $K_4$ -subdivision in graphs, in “*Surveys in Combinatorics, 1999 (London)*”, London Math. Soc. Lecture Note Ser. **267**, Cambridge Univ. Press, Cambridge, 1999, 223–238.
- [61] C. Thomassen, Totally odd  $K_4$ -subdivisions in 4-chromatic graphs. *Combinatorica* **21** (2001), 417–443.
- [62] B. Toft, Problems 10 and 11, in *Recent advances in graph theory: Proceedings of the Symposium held in Prague, June 1974* (ed. M. Fielder), Academia Praha, Prague (1975), 543–544.
- [63] W. Zang, Proof of Toft’s conjecture: Every graph containing no fully odd  $K_4$  is 3-colorable, *J. Combin. Optimization*, **2** (1998), 117–188.