

A Nearly Linear Time Algorithm For The Half Integral Parity Disjoint Paths Packing Problem

Ken-ichi Kawarabayashi^{*†‡}

Bruce Reed^{§¶}

Abstract

We consider the following problem, which is called the *half integral parity disjoint paths packing problem*.

Input: A graph G , k pair of vertices $(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)$ in G (which are sometimes called *terminals*), and a parity l_i for each i with $1 \leq i \leq k$, where $l_i = 0$ or 1 .

Output : Paths P_1, \dots, P_k in G such that P_i joins s_i and t_i for $i = 1, 2, \dots, k$ and parity of length of the path P_i is l_i , i.e, if $l_i = 0$, then length of P_i is even, and if $l_i = 1$, then length of P_i is odd for $i = 1, 2, \dots, k$.

In addition, each vertex is on at most two of these paths.

We present an $O(m\alpha(m, n)\log n)$ algorithm for fixed k , where n, m are the number of vertices and the number of edges, respectively, and the function $\alpha(m, n)$ is the inverse of the Ackermann function (see by Tarjan [43]). This is the first polynomial time algorithm for this problem, and generalizes polynomial time algorithms by Kleinberg [23] and Kawarabayashi and Reed [20], respectively, for the half integral disjoint paths packing problem, i.e., without the parity requirement.

As with the Robertson-Seymour 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. Here, however, we must maintain the parity of the paths and can only use an "odd clique minor". We must also describe the structure of those graphs in which we cannot find such a minor and discuss how to exploit it.

We also have algorithms running in $O(m^{1+\varepsilon})$ time for any $\varepsilon > 0$ for this problem, if k is up to $o(\log \log \log n)$ for general graphs, up to $o(\log \log n)$ for

planar graphs, and up to $o(\log \log n/g)$ for graphs on the surface, where g is Euler genus. Furthermore, if k is fixed, then we have linear time algorithms for the planar case and for the bounded genus case.

1 Introduction

In the vertex- (edge-) disjoint paths problem, we are given a graph G and a set of k pairs of vertices in G , and we have to decide whether or not G has k vertex-(edge-) disjoint paths connecting given pairs of terminals. This is certainly a central problem in algorithmic graph theory and combinatorial optimization. See the surveys [11, 40]. It has attracted attention in the contexts of transportation networks, VLSI layout and virtual circuit routing in high-speed networks or internet.

Let us give previous known results on the disjoint paths problem. If k is as a part of the input of the problem, then this is one of Karp's original NP-complete problems [22], and it remains NP-complete even if G is constrained to be planar (Lynch [27]). The seminal work of Robertson and Seymour says that there is a polynomial time algorithm (actually $O(n^3)$ algorithm) for the disjoint paths problem when the number of terminals, k , is fixed. Actually, this algorithm is one of the spin-offs of their groundbreaking work on Graph Minor project, spanning 23 papers, and giving several deep and profound results in Discrete Mathematics. Recent improvements are given in [21].

In the multi-commodity flow question, the commodities at the sources s_1, \dots, s_k are different and the demand at each t_i is for a specific commodity. This is the type of question we need to resolve when sending information through the information highway network and so has become increasingly of interest to computer scientists (see, for example the work of Chekuri et al. [4, 5] and of Tardos and Kleiberg [23, 24]). One special case which is of great interest is that all demands are at most $1/2$. This problem setting behaves very different from the disjoint paths problem. Indeed there are many such flow type problems for which the half integral version can be at least approximately solved although the integral version is intractable ([25, 28]). A similar situation holds with respect to the k -disjoint path problem. The proof of correctness of Robertson and Seymour's algorithm requires almost all of the graph minors project spanning 22 papers and more than 500 pages. Its running time has the form $O(f(k)n^3)$, where f is an extremely rapidly growing function. On the other hand,

^{*}National Institute of Informatics, 2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo, Japan.

[†]Research partly supported by JSPS Postdoctoral Fellowship for Research Abroad.

[‡]Email address: k_keniti@nii.ac.jp

[§]Canada Research Chair in Graph Theory, McGill University, Montreal Canada

[¶]Email address: breed@cs.mcgill.ca

Kawarabayashi and Reed [20] gave a nearly linear time algorithm for the half integral version.

THEOREM 1.1. *There is an $O(n \log n)$ time algorithm for the following problem for fixed k , which is called the half-integral k disjoint paths packing problem.*

Input: A graph G , k pair of vertices $(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)$ in G (which are sometimes called terminals, and are not necessarily disjoint).

Output : Paths P_1, \dots, P_k in G such that P_i joins s_i and t_i for $i = 1, 2, \dots, k$, and in addition, each vertex (except possibly for the terminals) is on at most two of these paths.

This improves the previous known result by Kleinberg [23] who gave an $O(n^3)$ algorithm. In addition, the correctness of this algorithm is much simpler than that of Robertson and Seymour's, and the hidden constant is much smaller.

Let us remark that the half-integral disjoint paths packing problem is still NP-complete if k is as a part of input, see [28].

Our motivation is to extend Theorem 1.1 to the parity version.

1.1 The half-integral parity disjoint paths packing problem. The half-integral parity disjoint paths packing problem (HFDP) is the following.

Input: A graph G , k pair of vertices $(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)$ in G (which are sometimes called "terminals", and are not necessarily disjoint), and a parity l_i for each i with $1 \leq i \leq k$, where $l_i = 0$ or 1 .

Output : Paths P_1, \dots, P_k in G such that P_i joins s_i and t_i , and parity of length of the path P_i is l_i , i.e, if $l_i = 0$, then length of P_i is even, and if $l_i = 1$, then length of P_i is odd for $i = 1, 2, \dots, k$. In addition, each vertex is on at most two of these paths.

Recently, many researchers are interested in the parity version of the half-integral disjoint paths problem and the integral disjoint paths problem. This is because, they are connected to some deep theory and results. I.e, the odd S -paths theorem [6, 12] (which generalizes the well-known Mader's S -paths theorem), the odd clique minor and the generalized Hadwiger's conjecture [12, 15, 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 [42]), and the odd disjoint cycles problem [18, 42]. In fact, the odd disjoint cycles problem is one of the central combinatorial optimization problems, since it is connected to Matroid theory, Matching Theory, and Structure Graph Theory.

As discussed above, there is some recent progress in this area. Our goal is to solve the parity version of the disjoint paths problem (for fixed number of terminals). However, this needs a significant generalization of the Robertson-Seymour's algorithmic framework. We have

made progress on this project, and we found that the solution would need > 200 pages proof, which essentially extends all the Graph Minors work to the parity version. So far, we do not finish writing all the proofs yet, but we believe we have done all the needed work. While working on this problem, we found that the half-integral version of the parity disjoint paths problem behaves very different. As with the Kleinberg's algorithm [23] and the Kawarabayashi-Reed's algorithm [20] for the half-integral version of the disjoint paths problem, we can get a shorter and simpler proof for the half-integral version of the parity disjoint paths problem (though we still need a 30 pages proof, which is much longer than the proof of the half-integral disjoint paths problem [20, 23]). The purpose of this paper is to present this result. In fact, we can adapt the approach by Kawarabayashi and Reed [20] to get a nearly linear time algorithm for the problem.

The half-integral parity disjoint paths packing problem clearly implies the half-integral disjoint paths packing problem, because if we would test the algorithm of the half-integral parity disjoint paths packing problem at most 2^k times (for all possible parties of P_i), then clearly we would get an answer of the half-integral disjoint paths packing problem. If k is part of the input, it is clearly NP-complete. The question is; what if the number of terminals, k , is fixed? This is the motivation of our research, and our main result gives the first polynomial time algorithm (actually, the time complexity is almost linear) for this problem.

To appreciate why the parity version of the problem is much harder than the nonparity version, it is useful to consider the following related concepts: namely, Erdős-Pósa property and odd disjoint cycles.

1.2 Erdős-Pósa property, odd disjoint cycles and difficulty for the parity paths problem. A family \mathcal{F} of graphs is said to have the *Erdős-Pósa property*, if for every integer k there is an integer $f(k, \mathcal{F})$ such that every graph G contains k vertex-disjoint subgraphs each isomorphic to a graph in \mathcal{F} or a set C of at most $f(k, \mathcal{F})$ vertices such that $G - C$ has no subgraph isomorphic to a graph in \mathcal{F} . The term *Erdős-Pósa property* arose because in [9], Erdős and Pósa proved that the family of cycles has this property.

On the other hand, for odd cycles, the situation is different. The Erdős-Pósa property does not hold for odd cycles in general. If we take a projective planar graph such that each face has an even size, and all the non-contractible cycles have odd length, then it does not satisfy the property, as remarked in [29]. However, Reed [33] proved that the Erdős-Pósa property holds for odd cycles in planar graphs. This result was extended to an orientable fixed surface in [19].

The difficulty for testing k disjoint odd cycles is that, since the Erdős-Pósa property does not hold for odd cycles (even $k = 2$), so, we cannot reduce the problem to the disjoint paths problem in bipartite graphs. If the Erdős-Pósa property would hold for odd cycles, then we could just guess a vertex set X that

has bounded number of vertices such that $G - X$ is bipartite. Because every odd cycle would pass through at least one vertex of X , and there are at most $|X|^k$ ways for k disjoint odd cycles to pass through the vertex set X , so we could use the algorithm of Robertson and Seymour with respect to the terminals in X , and test whether or not there are k disjoint odd cycles, since we do not have to worry about parity of disjoint paths in the bipartite subgraph $G - X$. But this argument does not work simply because the Erdős-Pósa property does not hold for odd cycles (even $k = 2$). This also suggests that the parity version of the disjoint paths problem is much harder than the nonparity version.

But for the half-integral version of the disjoint odd cycles problem, Reed [33] proved that the Erdős-Pósa property does hold. Consequently, Reed's result [33] implies the following.

THEOREM 1.2. *For fixed k , there is a polynomial time algorithm for testing whether or not there is a half-integral k disjoint odd cycles packing.*

Later, Reed, Smith and Vella [36] gave an $O(nm)$ algorithm by using Reed's theorem in [33].

Our second motivation is to extend this result to the half-integral parity disjoint paths packing problem.

The Robertson-Seymour's algorithmic framework is not enough to solve this problem, since we need to control the parity of the paths. More precisely, as with the Robertson-Seymour 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. Here, however, we must maintain the parity of the paths and can only use an "odd clique minor". We must also describe the structure of those graphs in which we cannot find such a minor and discuss how to exploit it. This motivates us to introduce an odd huge clique minor to attack this problem.

In order to obtain a huge odd clique minor, we need some recent deep results concerning odd S -paths, and an odd clique minor, see [6, 12]. We also need to exploit the structure of graphs without a huge odd clique minor. This also needs more work. In addition, we need some profound results in Graph Minors. Thus the half-integral parity disjoint paths packing problem is much harder than the half-integral disjoint paths packing problem.

1.3 Our Main Result. We prove the following result. We denote m, n by the number of edges and vertices, respectively. The function $\alpha(m, n)$ is the inverse of the Ackermann function, as observed by Tarjan [43].

THEOREM 1.3. *There is an $O(m\alpha(m, n)\log n)$ algorithm for the half-integral parity disjoint paths packing problem for fixed k , that is, the number of terminals is fixed.*

If we do not care about the time complexity in Theorem 1.3, then we could follow our approach in

this paper, and then use all known results from Graph Minors [38], together with apparently new algorithmic results for odd S -paths and odd clique minors [6, 12], to get an $O(n^3)$ algorithm (although as far as we know, no one could come up with this result). In fact, if we only require an $O(n^3)$ algorithm for this problem, we could obtain a little shorter proof (but it would only save a few pages). But we would rather adapt our recent progress on our algorithmic graph minor theory [20, 21] to get a much faster algorithm, which is close to linear. This method also allows us to give a much better hidden constant. As with Kleinberg's algorithm, and Kawarabayashi-Reed's algorithm, our algorithm can also handle k which grow slowly with n .

THEOREM 1.4. *There is an algorithm running in $O(m^{(1+\varepsilon)})$ time for any ε to decide the feasibility of the half-integral parity disjoint paths packing problem on an arbitrary n -vertex graph with $k \leq o(\log \log \log n)$.*

The approximation ratio " $o(\log \log \log n)$ " depends on the bound of the tree-width $f(k)$ that forces a $k \times k$ grid minor. Currently, the best known bound is 20^{2k^5} by [41]. But if a given graph is planar, then the situation is different. Robertson, Seymour and Thomas [41] proved that any planar graph with tree-width at least $6k$ has a $k \times k$ -grid minor. This together with our method implies the following much better result.

THEOREM 1.5. *There is an algorithm running in $O(n^{(1+\varepsilon)})$ time for any ε to decide the feasibility of the half-integral parity disjoint paths packing problem on an arbitrary n -vertex planar graph with $k \leq o(\log \log n)$. In addition, if k is fixed, then the running time is $O(n \log n)$.*

We can also give a similar result for graphs on a fixed surface, i.e., a bounded genus graph. Let g be Euler genus of the surface, and suppose G is embedded on this surface. It was proved that such a graph with tree-width at least $6kg$ has a $k \times k$ -grid minor (see [7]). Using this result, we can obtain the following theorem.

THEOREM 1.6. *There is an algorithm running in $O(n^{(1+\varepsilon)})$ time for any ε to decide the feasibility of the half-integral parity disjoint paths packing problem on an arbitrary n -vertex graph that is embedded on a fixed surface with Euler genus g and with $k \leq o(\log \log n/g)$. In addition, if k is fixed, then the running time is $O(n \log n)$.*

Reed et al. [35] proved that for planar graphs, there is a linear time algorithm for the disjoint paths problem for fixed k . As discussed in Reed [34], they also obtained a linear time algorithm to solve this problem on graphs embedded on a fixed surface. Combining these techniques and results, we can actually give linear time algorithms for Theorems 1.5 and 1.6 for fixed k , respectively.

Our proof method for Theorem 1.3 can be used to design $O(m\alpha(m, n)\log n)$ algorithms for the following problems too.

1. Not only half-integral, but also that all demands on vertices have at most $1/2$.
2. Not only vertex capacities problem, but also edge capacities as long as all demands on edges are at most $1/2$.

Since our proof can be easily modified for these problems, so we omit proofs.

1.4 Overview of the proof. At a very high level, we shall follow the approach adapted by Kawarabayashi and Reed [20]. Let us give a sketch of the algorithm first. We adopt a divide and conquer approach. The approach to the half-integral disjoint paths packing is to attempt to construct a tree decomposition of bounded tree-width of a graph obtained from G by repeatedly applying some reductions. We can then use dynamic programming to solve the half-integral disjoint paths packing on it in linear time.

Let us construct a tree-decomposition. Specifically, we will have a bound $6B$ on $W_s \cap W_t$ for adjacent nodes s and t of T and each W_t has at most $8B$ vertices, where B will be determined later. The only reason we fail to construct the tree decomposition is because the tree-width is not small. Then there is a wall of huge height by the result in [8, 32, 37, 41]. Now, we shall use this wall of huge height as a "crossbar structure". It turns out that given a big enough wall for which there is no small cut separating the terminals from it, we can find the desired paths. This was already proved by Kleinberg [23]. Even if there is a small cutset that separates the $2k$ terminals from this wall of huge height, we will be able to reduce the problem. This was proved in [20]. So once we have a wall of huge height, either we can make a smaller graph or we can conclude that the desired half-integral disjoint paths packing exists.

But when we make a reduction, we need to throw away at least half vertices. In order to do so, we cannot just apply a reduction to any wall. We need to apply a reduction to a wall which is "attached" to a specific highly connected part of the graph. This can be done by Reed's result [31], Theorem 2.2, together with a related algorithmic result, see in [20, 21], and Theorem 2.3. With a few technical complications, we can do this reduction for all leaves simultaneously. This allows us to construct a tree-decomposition of bounded width in $O(n \log n)$ time.

Our approach to the half-integral parity disjoint paths packing problem is similar, but there are two huge differences:

1. We attempt to construct a tree-decomposition, but not necessarily of bounded tree-width. Specifically, either each piece has bounded order, or it consists of 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) such that after deleting X from the piece, the resulting graph is bipartite.
2. We also make a reduction, but we need to consider not only a wall of large height, but also a huge odd-

clique minor, a huge clique minor without huge odd clique minors, an "even" wall of large height with many "parity" breaking paths, and so on. We shall discuss more details later.

These two differences occurs when we consider a "reduction" for our problem. In the half-integral disjoint paths packing problem, we can always make a reduction by using a wall of huge height, which is "attached" to a specific highly connected part of the graph. This allows us to keep constructing a desired tree-decomposition. But in our problem, the reduction is much more difficult. In fact, in some cases, we need the structure as in 1, i.e., some of the pieces become nearly bipartite graphs.

We need a new concept, "odd-minor". We need to address what it is.

Recall that a graph H is a *minor* of G if H can be obtained by contracting and deleting edges in G . 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) is more general than the class of H -minor-free graphs (excluding a fixed graph H as a minor).

Indeed, the class of odd- H -minor-free graphs is strictly more general: 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 $k \geq 3$. Also, any K_k -minor-free graph G is $O(k\sqrt{\log k})$ -degenerate, i.e, every induced subgraph has a vertex of degree at most $O(k\sqrt{\log k})$; see [26, 44]. 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.

Let us turn back to overview of our main result. We either make a reduction or continue to construct the tree-decomposition. To do so, we need to consider the following three cases.

1. A huge odd clique minor with a huge clique minor.
2. A huge clique minor, but no huge odd clique minors.
3. A wall of huge height, but no huge clique minor.

More precisely, the third case needs the following two subcases.

- (a) A flat wall of huge height with many disjoint odd faces.
- (b) A bipartite flat wall of huge height, with many disjoint "parity breaking" paths.

We now remark that all the above configurations are “attached” to a specific highly connected part of the graph. This allows us to throw away almost half of the vertices when we perform a reduction.

Suppose there is a huge clique minor. Robertson and Seymour also excluded a huge clique minor for the k disjoint paths problem, and that step is not hard. But our problem setting is much harder since we need to figure out a parity of each path. To do so, we first need to exclude a huge odd clique minor. So this corresponds to the first case. We will prove that if there is a huge odd clique minor, then either there are desired paths, or there is a “reduction”, using the odd clique minor. Note that this odd clique minor we shall get is “attached” to a specific highly connected part, so when we perform a reduction, we can throw away almost half of the vertices.

One question is: how do we find a huge clique odd minor? It turns out that we have a nice structure theorem which tells us that if we have a huge clique minor, then either we can get a big odd minor or else we can get a vertex set X of bounded size (depending on k) such that the component of $G - X$ containing most of the nodes of the huge clique minor is “essentially” bipartite. This essentially follows from the result in [12].

Then we need to discuss the case when there is a huge clique minor, but no huge odd clique minors. So this corresponds to the second case. In this case, by the above result, we can conclude that the graph which is “attached” to a specific highly connected part is “essentially” bipartite, and this negates our need to construct the tree-decomposition.

We now come to the point where there is no huge clique minor. By the result of Robertson and Seymour [38], there is a huge flat wall W , which is “attached” to a specific highly connected part of the graph. In fact, we can find such a wall in linear time by the recent result in [21]. So this corresponds to the third case, but we need to look at the cases (a) and (b).

If W has many disjoint odd faces, then by the similar argument in Kawarabayashi and Reed [16], either desired paths exist, or we can make a reduction, and throw away half of the vertices. This corresponds to (a) of the third case.

So, we are left with the case that there is a huge flat subwall W' of the wall W , which is bipartite. So this corresponds to (b) of the third case. In this case, by the above result by Geelen et al. [12], we can conclude either that the graph “attached” to a specific highly connected part is “essentially” bipartite, (and hence in this case, this negates our need to construct the tree-decomposition, as discussed above), or there are many parity breaking paths, each joining two points on the outer cycle of the wall W' . By a *parity breaking path*, we mean a path P with endpoints on the outer cycle of the wall W' such that the path P together with the wall W' gives rise to an odd cycle. In the second case, by the similar argument in Kawarabayashi and Reed [16], either desired paths exist or we can make a reduction, and throw away half of the vertices.

With a few technical complications, we can do this reduction for 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. Let us remark that 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 of k . Finding an augmenting path takes $O(m\alpha(m, n))$. That is why $\alpha(m, n)$ in the time complexity comes in.

Given the tree decomposition constructed using the above approach, we want to solve the problem on it via dynamic programming working our way up from the leaves.

Let us remind that the tree-decomposition may not have bounded tree-width. More precisely, 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 $f(k)$ for some function of k . Also, $W_t \cap W_s \subseteq X$, where s is the parent of t in the corresponding tree T . $W_s \cap W_t$ is the root for W_t .

We now work our way up from the leaves of the tree-decomposition. In order to do so, we need to solve rooted versions of the problem (the roots corresponding to the problem for a node t are $W_s \cap W_t$ for the parent s of t). More precisely, we need to solve the rooted problem at each node of the tree decomposition, given that we have done so for its children. Often, we can replace each child by a bounded size graph which is the smallest graph giving the same solution to the rooted problem.

In the above case 1, we just apply the bounded tree-width method. While in the above case 2, we apply Theorem 1.1 to replace by a bounded size graph. Let us observe that we only need to consider the non-parity version of the problem for nearly bipartite graphs, since $|X|$ is bounded and in the bipartite subgraph $W_t - X$, we do not have to worry about parities of half-integral disjoint paths packing. Moreover, we only need to consider the case that all the terminals are in X , and hence the number of the terminals is bounded.

In this way, we can solve the half-integral parity disjoint paths problem by using dynamic programming.

2 Preliminary

A separation (A, B) is that $G = A \cup B$, and there are no edges between $A - B$ and $B - A$. The order of the separation (A, B) is $|A \cap B|$.

We will omit the notation “bramble, gird minor and wall”, as they are now standard. One of the most important results concerning the tree-width is the main result of Graph Minor V [37] which says the following.

THEOREM 2.1. *For any r , there exists a constant $f(r)$ such that if G has tree-width at least $f(r)$ (equivalently a bramble B of order at least $f(r)$), then G contains a wall W of height r .*

The best known upper bound for $f(r)$ is given in [8, 32, 41]. It is 2^{205^r} . The best known lower bound is

$\Theta(r^2 \log r)$, see [41]. For $r = 3, 4$, this upper bound is improved by Birmel e, Bondy and Reed [2].

Reed [31] gave an $O(n \log n)$ algorithm to construct a tree decomposition of width at most $4k$ for a graph of tree width k . In his proof, the following theorem, which we find very important for our purpose, was proved.

THEOREM 2.2. (REED, [31]) *For any fixed integer k , there is a linear time algorithm which, given a graph G , either:*

1. *finds a cutset Y of G with $|Y| \leq k$ such that no component of $G - Y$ contains more than $\frac{2}{3}|G - Y|$ vertices, or*
2. *determines that for any set Y of vertices of G with $|Y| \leq k$, there is a component of $G - Y$ which contains more than $\frac{1}{2}|G - Y|$ vertices.*

It is easy to see that if the outcome is 2, then we have determined that the bramble β_G , consisting of all the connected subgraphs containing more than half the vertices of G , has order at least $k/2$.

A related algorithmic result of Theorem 2.1 was proved by Reed, see in [20, 21]. Namely:

THEOREM 2.3. *For any k , there is a function $f(k)$ such that, for any $D \geq f(k)$, there is a linear time algorithm, given any bramble β_X of order $f(k)$ for some vertex set X which either has size at most D or is $V(G)$, to construct a wall W of height k , which is controlled by β_X , i.e., for every subset Y of small order (say order less than k), the unique component of $G - Y$ containing a path from the top row of W to the bottom row of W contains more than half the vertices of X . It follows that the component contains an element of the bramble β_X .*

The function $f(k)$ in Theorem 2.3 is almost same as the function $f(r)$ in Theorem 2.1.

We need to make clear how the bramble in Theorem 2.3 will be given. If we perform Theorem 2.2 and the conclusion is 2, then this implies that the bramble β_X for some small set X or $X = V(G)$ has order at least $k/2$. This bramble is one of keys for our algorithm, and will be used in Theorem 2.3.

Let us remark that if we say that a minor H is controlled by the bramble β_G , then this means that, for any small cutset Z (say order less than $|H|/2$), the unique component of $G - Z$ containing at least $|H|/2$ nodes of the H -minor also contains more than half of the vertices of G . It follows that the component contains an element of the bramble β_G .

We shall use the following recent result in [12].

THEOREM 2.4. *For any set S of vertices of a graph G , and for any fixed k , 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 [12] actually implies that there is a polynomial time algorithm for giving one of the conclusion in Theorem 2.4.

In general, this algorithm takes $O(km\alpha(m, n))$ time, where the function $\alpha(m, n)$ is the inverse of the Ackermann function, as observed by Tarjan [43], because we need to find an augmenting path for k times, and finding each augmenting path takes $O(m\alpha(m, n))$. In our case, k is fixed. So we can find either 1 or 2 in Theorem 2.4 in nearly linear time.

3 Algorithm: Constructing a tree-decomposition

Reed [31] gave an $O(n \log n)$ algorithm to construct a tree decomposition of width at most $4k$ for a graph of tree width k . Extending his approach, we attempt to construct a tree decomposition of a reduction of the input graph for which the solution to the half-integral parity disjoint paths packing problem is the same as for the original input graph. Each piece of this tree decomposition is either a graph of bounded size, or a nearly bipartite graph. If we fail to do so, we actually find the desired paths.

Specifically, we construct the decomposition iteratively. In every iteration, we construct a tree decomposition each of whose internal nodes either is nearly bipartite (i.e., it has a vertex set X of at most $8B$ vertices that contain all the vertices of the root such that after deleting X from it, the resulting graph is bipartite. Hereafter, when we refer "nearly bipartite graph", we mean the graph described here) or has bounded number of vertices (at most $8B$), and such that the bound $6B$ on $|W_s \cap W_t|$ holds for all edges st of T , where s is the parent of t .

Actually, there are two kind of iterations which we shall alternate. One is concerning splitting roots of each leaf, and the other is concerning splitting the vertices of each leaf. Here, by a root R^t of a leaf t of the tree T , we mean that $W_s \cap W_t$ for the edge st of T , where s is the parent of t . In each $2i - 1$ th iteration, we extend the tree decomposition each of whose internal nodes either is nearly bipartite or has bounded number of vertices (at most $8B$), and such that the bound $6B$ on $|W_s \cap W_t|$ holds for all edges st , where neither s nor t is a leaf, and in addition, the bound $5B$ on $|W_s \cap W_t|$ holds for all edges st , where t is a leaf of T , by splitting root of each leaf. In each $2i$ th iteration, we extend the tree-decomposition by splitting each leaf into half such that the bound $6B$ on $|W_s \cap W_t|$ holds for all edges st of T . Then the size of the leaf of the new decomposition is now at most half of the leaf of the previous one. In the $2i - 1$ th iteration, the maximum size of W_t for a leaf t is $\max(|V|^{3/4}, 8B)$. It follows that after $4 \log n$ iterations, each such leaf node is of size at most $8B$.

In each iteration there are two possibilities. Either the bramble β_X for the set X of vertices we are trying to split has large order or it has small order. In the

second case, it is easy to carry out the desired split. In the first case, we perform a reduction, using this high order bramble β_X .

We are now ready to describe our algorithm. More details can be found in the full version of this paper.

As we pointed out, we shall alternate two iterations. Each iteration has two phases as we saw two possibilities above.

In the very first iteration, we choose a set of $5B$ roots arbitrarily subject to the condition that they contain all the $2k$ terminals T for the half-integral parity disjoint paths packing problem.

Iteration 1: Splitting Roots

For this iteration, the input is a tree decomposition each of whose internal nodes either is nearly bipartite or has bounded number of vertices (at most $8B$), and such that the bound $6B$ on $|W_s \cap W_t|$ holds for all edges st . We assume momentarily that there is only one leaf t such that W_t has size bigger than $8B$.

Then we output either desired paths, or a new tree-decomposition that is a refinement of the old tree-decomposition, each of whose internal nodes either is nearly bipartite or has bounded number of vertices (at most $8B$), and such that the bound $6B$ on $|W_s \cap W_t|$ holds for all edges st , where neither s nor t is a leaf of T , and in addition, the bound $5B$ on $|W_s \cap W_t|$ holds for all edges st , where t is a leaf of T .

Here is a description. This iteration has two phases: either (a) we split the roots of W_t , or (b) we perform a reduction. If we succeed (a), then we output the result and go to the Iteration 2. Otherwise we go to (b). The phase (b) allows us to delete half of R^t , which negates our need to split the roots into smaller sized pieces.

In the phase (a), we first attempt to find a set X of at most B vertices such that no component of $G(W_t) - X$ contains more than two-thirds of the vertices of R^t . Note that there is a partition of W_t into two pieces so that each piece has at most two-thirds of the vertices of the root R^t if and only if there is no piece with more than two-thirds of the vertices of the root R^t . This is tight, since there may be a partition of W_t into three pieces so that each component has exactly one-thirds of the vertices of the root R^t .

We can either do so or determine that no such set exists in linear time by computing the following: For every partition of the root R^t of the leaf W_t into A , D and C , figure out whether or not in W_t , there is a cutset R of size at most B that separates A from D whose intersection with the roots is C . If there is, we detect a smallest cutset. This can be clearly done in time $O(|W_t|)$, since there are at most $6B$ vertices in R_t . So we have only $3^{6|R|}$ choices for A, C, D , and for each choice, we just perform a B disjoint $a - b$ paths algorithm for fixed value B , which can be done in linear time. See [13].

If we succeed then we replace t in T with a star with center t^* and one leaf for each component of $G(W_t) - X$. There will be an edge of our new tree decomposition between t^* and each leaf l . W_{t^*} will be $R^t \cup X$. This is

small enough that W_{t^*} has at most $8B$ vertices. For the leaf l of the star corresponding to some component U of $G(W_t) - X$, W_l will be $U \cup X$ and so $W_l \cap W_{t^*}$ will be $X \cup (R^t \cap U)$ which has size at most $5B$ by our choice of X . Then the phase (a) is done. So we output the result and go to the Iteration 2. Otherwise, we perform the phase (b) below. (Actually, we make the leaf "Robust", i.e, for any two vertex sets A, C of R^l of order at most $|W_l \cap W_{t^*}|/2$ with $|A| \leq |C|$, there are $|A|$ disjoint paths between A and C in $(W_l - (W_l \cap W_{t^*})) \cup (A \cup B)$. This can be done by repeatedly (but at most $6B$ times) taking a smallest separation X in the above argument.)

Now we perform the phase (b), which makes a reduction. If we fail to find the desired set X in the phase (a), then the bramble β_{R^t} has order at least B . We then apply a linear time algorithm which finds a large wall M (say a wall of height r , where $r \geq 2^{8k}$), controlled by this high order bramble β_{R^t} (as we are free to make B as large as we like), as described in Theorem 2.3. Let us observe that for any small cutset Z of size at most 2^k , the unique component of $G - Z$ containing a path from the top row of M to the bottom row of M also contains more than half of the vertices of R^t .

We then figure out the following:

1. Either W_t has a huge clique minor of order l (say $l \geq k^2$), controlled by the bramble β_{R_t} or
2. W_t has a flat subwall M' (of the wall M) of height 2^{4k} , obtained by deleting bounded number of vertices X in W_t . Moreover M' is controlled by the bramble β_{R_t} .

We can detect one of them in linear time (More details can be found in the full version).

Suppose the case 1 applies. We first attempt to find a huge odd clique minor (say order $7k$) by using the odd S -path theorem. In time $O(m\alpha(m, n))$ time, we can either get a huge odd clique minor (say order $7k$), controlled by the bramble β_{R_t} , or there is a vertex set X' of order at most $56k$ in such a way that the unique component of $W_t - X'$ containing at least $k^2 - 56k$ nodes of the clique minor, can be written as W, B_1, \dots such that W is bipartite, and each B_i contains an odd cycle and is a block with a cutvertex shared with a vertex in W . In addition, $W_t - X' - B_i$ contains more than half of the vertices of R^t for each i (More details can be found in the full version).

If we get a huge odd clique minor, then we can either get the desired half-integral parity disjoint paths packing, or make a reduction. Note that when we make a reduction, we shall throw away half of the roots of the leaf, since the odd clique minor is controlled by the bramble β_{R^t} , and hence the part we shall throw away contains at least half of the roots of the leaf (More details can be found in the full version).

Otherwise, we can conclude the second outcome. We assume that $B \geq |X'| + 1$. Then each component of $W_t - X'$ not containing the bipartite graph W , together with X' , and each B_i , together with X' , negate our need to split the roots of the leaves, since they become

leaves. More precisely, we replace t in T with a star with center t^* , which corresponds to the graph $X' \cup W \cup R_t$. Note that the graph $X' \cup W \cup R_t$ is nearly bipartite, and by our choice of X' , $X' \cup R_t$ contains at most $8B$ vertices. Each leaf of the star corresponds to either a component in $W_t - X'$ not containing the bipartite graph W , together with X' , or $B_i \cup X'$ for each i . There will be an edge of our new tree decomposition between t^* and each leaf l . For the leaf l of the star, $W_l \cap W_{t^*}$ will be $X' \cup (R_t \cap W_l)$ which has size at most $5B$ since each B_i contains at most $3B$ of R_t , and thus $B_i \cup X'$ has a root with at most $3B + |X'| + 1 < 5B$ vertices for each i . Let us observe that every component of $W_t - X'$ not containing the bipartite graph W has at most half of the vertices of R^t (More details can be found in the full version).

Suppose the case 2 applies. Let us observe that for any small cutset Z of order at most 2^k , the unique component of $G - Z$ containing at least 2^k vertices of the wall M' , which have degree ≥ 3 in M' , also contains more than half of the roots of R^t . If M' has many disjoint odd faces, each of which is far apart from any other, then we can either get the desired half-integral parity disjoint paths packing, or make a reduction. Note that when we make a reduction, we shall throw away half of the roots of the leaf, since the wall M' is controlled by the bramble β_{R^t} , and thus the part we shall throw away contains at least half of the roots of the leaf (More details can be found in the full version).

Otherwise, we can conclude that there is a flat subwall M'' (of the wall M') of height r' , where $r' \geq 2^{2k+1}$, such that M'' has no odd faces, and thus is bipartite. Take all the vertices of degree ≥ 3 (in M'') in the outer face boundary of M'' , which are in one partite set of bipartitions of M'' . Let them S . If there are k^2 disjoint odd paths in $W_t - X$ such that each of the endpoints is on S , then clearly these paths do not hit any vertex of M'' except for its outer cycle, since M'' is a flat wall in $W_t - X$. By shrinking the wall M'' (if necessary), we can clearly take a flat subwall M_1 (of the wall M'') of height $r'/2$ with k disjoint odd paths such that each of these paths connects two vertices (of degree ≥ 3 in M_1) of the outer cycle of M_1 , and does not hit any vertex of M_1 except for its outer cycle. Moreover, each of the endpoints is distance at least k^2 from any other. Then we can either get the desired half-integral parity disjoint paths packing, or make a reduction, since this configuration is enough for our purpose. Note that when we make a reduction, we shall throw away half of the roots of the leaf, since the wall M_1 is controlled by the bramble β_{R^t} , and thus the part we shall throw away contains at least half of the roots of the leaf (More details can be found in the full version).

Otherwise, by Theorem 2.4, it follows that there is a vertex set X' of order 2^k in such a way that the unique component of $W_t - X' - X$ containing at least 2^k vertices of the wall M'' , which have degree ≥ 3 in M'' , can be written as W, B_1, \dots such that W is bipartite, and each B_i contains an odd cycle and is a block with a cutvertex shared with a vertex in W . In addition,

$W_t - X' - B_i$ contains more than half of the vertices of R^t for each i . We assume that $B \geq |X'| + |X| + 1$. Then each component of $W_t - X' - X$ not containing the bipartite graph W , together with $X \cup X'$, and each B_i , together with $X \cup X'$, negate our need to split the roots of the leaves since they become leaves. More precisely, we replace t in T with a star with center t^* , which corresponds to the graph $X' \cup W \cup R_t \cup X$. Note that the graph $X' \cup W \cup R_t \cup X$ is nearly bipartite, and by our choices of X and X' , $X' \cup X \cup R_t$ contains at most $8B$ vertices. Each leaf of the star corresponds to either a component in $W_t - X' - X$ not containing the bipartite graph W , together with $X' \cup X$, or $B_i \cup X' \cup X$ for each i . There will be an edge of our new tree decomposition between t^* and each leaf l . For the leaf l of the star, $W_l \cap W_{t^*}$ will be $X' \cup (R_t \cap W_l) \cup X$ which has size at most $5B$ since each B_i contains at most $3B$ of R_t , and thus $B_i \cup X' \cup X$ has a root with at most $3B + |X'| + |X| + 1 < 5B$ vertices for each i . Let us observe that every component of $W_t - X' - X$ not containing the bipartite graph W has at most half of the vertices of R^t .

In summary, either desired paths exist, or we can continue the tree-decomposition, or we can make a reduction. Moreover, when we make a reduction, we can actually delete half of R^t , which negates our need to find the vertex set X to split the roots as we used it to split the roots into smaller sized pieces. Hence our purpose to make the root half of the original R^t is already achieved. This finishes the phase (b), and we output it. Then go to the Iteration 2. Let us remark that 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 of k . Finding an augmenting path takes $O(m\alpha(m, n))$. That is why $\alpha(m, n)$ in the time complexity comes in.

Iteration 2: Splitting Each Leaf

After Iteration 1, we come to this iteration.

For this iteration, the input is a tree decomposition each of whose internal nodes either is nearly bipartite or has bounded number of vertices (at most $8B$), and such that the bound $6B$ on $|W_s \cap W_t|$ holds for all edges st , where neither s nor t is a leaf of T , and the bound $5B$ on $|W_s \cap W_t|$ holds for all edges st , where t is a leaf of T . Again, we assume momentarily that there is only one leaf t such that W_t has size bigger than $8B$.

Then we output either desired paths exist, or a new tree-decomposition that is a refinement of the old tree-decomposition, each of whose internal nodes either is nearly bipartite or has bounded number of vertices (at most $8B$), and such that the bound $6B$ on $|W_s \cap W_t|$ holds for all edges st .

Here is a description. Again, this iteration has two phases: either (c) we split the vertices of the leaf or (d) we make a reduction. If we succeed (c), then we output the result and go to the next Iteration 1. Otherwise we go to (d). The phase (d) allows us to throw away half of the vertices of W_t , which negates our need to split the leaves into smaller pieces.

In the phase (c), we attempt to find a set Y with at most B vertices such that no component of $G(W_t) - Y$

contains more than two-thirds of the vertices of W_t . We can either do so or determine that for every Z with at most B vertices, some component of $G(W_y) - Z$ contains more than half the vertices of W_t , in linear time, by the separator theorem, Theorem 2.2. If we find such a set Y then we proceed as in the Iteration 1 with Y in place of X . If this phase is successful, we are done. This phase ensures that our condition on the size of W_l for a leaf l holds. In addition, the number of roots for the new leaf is at most $6B$ vertices since after the Iteration 1, the number of roots for each leaf is at most $5B$ vertices. Then the phase (c) is done. So we output it and go to the next Iteration 1. Otherwise, we perform the phase (d) below.

Now we perform (d), which makes a reduction. If we fail, we proceed in a similar fashion in the phase (b), but considering β_{W_t} rather than β_{R^t} .

Much of the same occurs in the phase (d). Since the bramble β_{W_t} is huge enough, we then apply a linear time algorithm which finds a large wall M (say a wall of height r , where $r \geq 2^{8k}$), controlled by this high order bramble β_{W_t} (as we are free to make B as large as we like), as described in Theorem 2.3. As in the Iteration 1, this allows us either to find desired paths, or to continue the tree-decomposition, or to perform a reduction removing half the vertices of W_t . In the last case, this negates our need to find a set Y , as we used it to reduce the size of the W_t .

Let us observe that for any small cutset Z (of size at most 2^k), the unique component of $G - Z$ containing at least 2^k vertices of the wall M , which have degree ≥ 3 in M , also contains more than half the vertices of W^t by our choice of β_{W^t} . This allows us to throw away more than half of the vertices in W_t when we perform a reduction, since the part we shall throw away contains more than half of the vertices in W_t . Then go to the next Iteration 1 for the new leaves.

This completes the description of the Iteration 2. As in the Iteration 1, the Iteration 2 can be done in time $O(m'\alpha(m', n'))$, where m', n' are the number of edges and vertices of W_t , respectively.

In summary, we can do all the two iterations in time $O(m\alpha(m, n))$, and in each phase, either we can make a refinement of the tree-decomposition or reduce a size of our graph. Actually, in the second case, we throw away more than half of the vertices of the leaf. We shall do these two iterations for all leaves simultaneously. We can do it, except the only issue we need to deal with is that we can do all the reductions at once in linear time. We can do this because with a few technical complications we can ensure that the only interaction between the cuts is on the internal nodes of the tree decomposition, and, the tree-decomposition is now "Robust", so we can find all the cuts in linear time. This would allow us to construct a desired tree-decomposition in $O(m\alpha(m, n) \log n)$ time.

Let us remark that we take B large enough so that the bramble β_G of order at least B guarantees the existence of a wall of height at least 2^{8k} , controlled by this high order bramble β_G , as in Theorem 2.3.

So, in conclusion, we shall get a tree-decomposition such that either each piece is nearly bipartite or has bounded size in $O(m\alpha(m, n) \log n)$ time. Actually if the second case applies, then the size is at most $8B$ by our construction. Then we shall use Dynamic Programming approach. As pointed out in the overview, we want to solve problems on it via dynamic programming working our way up from the leaves. In order to do so, we need to solve rooted versions of the problem (the roots corresponding to the problem for a node t are $W_s \cap W_t$ for the parent s of t). Specifically, we need to solve the rooted problem at each node of the tree decomposition, given that we have done so for its children. Often, we can replace each child by a bounded size graph which is the smallest graph giving the same solution to the rooted problem.

One key point is that if W_t is nearly bipartite, we can apply Theorem 1.1 to replace by a bounded size graph. Let us observe that we only need to consider the non-parity version of the problem for nearly bipartite graphs, since $|X|$ is bounded and in the bipartite subgraphs $W_t - X$ we do not have to worry about parities of half-integral disjoint paths packing. Moreover, we only need to consider the case that all the terminals are in X , and hence the number of the terminals is bounded. If W_t has bounded size, then it is easy. We can then combine these two methods together. In this way, we can solve the half-integral parity disjoint paths problem by using dynamic programming. More details can be found in the full version.

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] E. Birmel e, J.A. Bondy and B. A. Reed, The Erdős-P osa property for long circuits, *preprint*.
- [3] T. B ohme, K. Kawarabayashi, J. Maharry and B. Mohar, Linear connectivity forces large complete bipartite minors, *to appear in J. Combin. Theory Ser. B*.
- [4] C. CHEKURI, S. KHANNA AND B. SHEPHERD, Edge-disjoint paths in planar graphs, Proc. 45th Ann. IEEE Symp. Found. Comp. Sci., 2004, pp. 71–80.
- [5] C. CHEKURI, S. KHANNA AND B. SHEPHERD, Edge-Disjoint Paths in Planar Graphs with Constant Congestion. 38th ACM Symposium on Theory of Computing (STOC), 2006.
- [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] 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.
- [8] 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.
- [9] P. Erdős and L. Pósa, On the maximal number of disjoint circuits of a graph, *Publ. Math. Debrecen* **9** (1962), 3–12.
- [10] S. Fortune, J.E. Hopcroft and J. Wyle, The directed subgraph homeomorphism problem, *Theor. Comput. Sci.* **10** (1980), 111–121.
- [11] A. Frank, Packing paths, cuts and circuits – a survey, in *Paths, Flows and VLSI-Layout* B. Korte, L. Lovász, H.J. Promel and A. Schrijver. Eds. Berlin: Springer-Verlag 1990, 49–100.
- [12] J. Geelen, B. Gerards, B. Reed, P. Seymour and A. Vetta, On the odd variant of Hadwiger’s conjecture, to appear in *J. Combin. Theory Ser. B*.
- [13] T. Ibaraki and H. Nagamichi, A linear time algorithm for finding a sparse k -connected spanning subgraph of a k -connected graph, *Algorithmica* **7**, (1992), 583–596.
- [14] K. Kawarabayashi, Improved algorithm for find a cycle through elements, *IPCO’08*, 374–384.
- [15] K. Kawarabayashi, Note on coloring graphs with no odd- K_k -minors, to appear in *J. Combin. Theory Ser. B*.
- [16] K. Kawarabayashi and B. Reed, Highly parity linked graphs, to appear in *Combinatorica*.
- [17] K. Kawarabayashi and Z. Song, Some remarks on the odd Hadwiger’s conjecture, *Combinatorica*. **27** (2007), 429–438.
- [18] K. Kawarabayashi and P. Wollan, Non-zero disjoint cycles in highly connected group-labelled graphs, *J. Combin. Theory Ser. B*. **96**, (2006), 296–301.
- [19] K. Kawarabayashi and A. Nakamoto, The Erdős-Pósa property for odd cycles on an orientable fixed surface, *Discrete Math.* **307** (2007), 764–768.
- [20] 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.
- [21] K. Kawarabayashi, Z. Li and B. Reed, Faster recognition and optimization algorithms for minor-closed graph family, *submitted*.
- [22] R. M. Karp, On the computational complexity of combinatorial problems, *Network*, **5**, (1975), 45–68.
- [23] J. Kleinberg, Decision algorithms for unsplittable flows and the half-disjoint paths problem, Proc. 30th ACM Symposium on Theory of Computing, 1998. 530–539.
- [24] J. Kleinberg, E. Tardos. Approximations for the disjoint paths problem in high-diameter planar networks. Proc. 27th ACM Symposium on Theory of Computing, 1995.
- [25] S. Kolliopoulos and C. Stein, Improved approximation algorithm for unsplittable flow problems, FOCS 1997.
- [26] A. Kostochka, Lower bound of the Hadwiger number of graphs by their average degree, *Combinatorica* **4** (1984), 307–316.
- [27] J.F. Lynch, The equivalence of theorem proving and the interconnection problem, *ACM SIGDA*, Newsletter, **5**, (1975), 31–65.
- [28] M. Middendorf and F. Pfeiffer, On the complexity of the disjoint paths problem, *Combinatorica*, **13** (1993), 97–107.
- [29] B. Mohar and C. Thomassen, Graphs on Surfaces, Johns Hopkins University Press, Baltimore, MD, 2001.
- [30] 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.
- [31] B. Reed, Finding approximate separators and computing tree width quickly; STOC 1992, Victoria B.C., 1992.
- [32] 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.
- [33] B. Reed, Mangoes and blueberries, *Combinatorica* **19** (1999), 267–296.
- [34] B. Reed, Rooted Routing in the Plane, *Discrete Applied Mathematics* **57**, 1995, 213–227.
- [35] B. Reed, N. Robertson, A. Schrijver and P. D. Seymour, Finding disjoint trees in planar graphs in linear time. Graph structure theory (Seattle, WA, 1991), 295–301, *Contemp. Math.*, **147**, Amer. Math. Soc., Providenc, RI, 1993.
- [36] B. Reed, K. Smith and A. Vetta, Finding odd cycle transversals, *Operation Research Letter* **32** (2004), 299–301.
- [37] N. Robertson and P. D. Seymour, Graph minors. V. Excluding a planar graph, *J. Combin. Theory Ser. B* **41** (1986), 92–114.
- [38] N. Robertson and P. D. Seymour, Graph minors. XIII. The disjoint paths problem, *J. Combin. Theory Ser. B* **63** (1995), 65–110.
- [39] N. Robertson and P. D. Seymour, Graph minors. XVI. Excluding a non-planar graph, *J. Combin. Theory Ser. B* **89** (2003), 43–76.
- [40] 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.
- [41] N. Robertson, P. D. Seymour and R. Thomas, Quickly excluding a planar graph, *J. Combin. Theory Ser. B* **62** (1994), 323–348.
- [42] P. Seymour, Matroid minors, *Handbook of Combinatorics*, (Eds.: R. L. Graham, M. Grötschel and L. Lovász). North-Holland, Amsterdam, 1985, 419–431.
- [43] R.E. Tarjan, Data Structures and network algorithms, SIAM, Philadelphia, PA, 1983.
- [44] A. Thomason, An extremal function for contractions of graphs, *Math. Proc. Cambridge Philos. Soc.* **95** (1984), 261–265.