

# Searching Paths of Constant Bandwidth

Bernd Borchert and Klaus Reinhardt

Universität Tübingen, Sand 13, 72076 Tübingen, Germany  
{borchert, reinhard}@informatik.uni-tuebingen.de

**Abstract.** As a generalization of paths, the notion of paths of bandwidth  $w$  is introduced. We show that, for constant  $w \geq 1$ , the corresponding search problem for such a path of length  $k$  in a given graph is NP-complete and fixed-parameter tractable (FPT) in the parameter  $k$ , like this is known for the special case  $w = 1$ , the LONGEST PATH problem. We state the FPT algorithm in terms of a guess and check protocol which uses witnesses of size polynomial in the parameter.

## 1 Introduction

A *path* within a graph is one of the most elementary notions of graph theory and its applications. The LONGEST PATH is the computational problem which asks for a given graph  $G$  and an integer  $k$  whether there is a path of length  $k$  in  $G$  which is simple, i.e. all vertices are different from each other. The LONGEST PATH is NP-complete [4]. Moreover, the LONGEST PATH problem is fixed-parameter tractable (FPT) in the parameter  $k$ . This was shown by Monien [5] and improved with respect to running time by Alon, Yuster, Zwick [1], using randomization techniques.

In this paper we generalize the notion of a path: a path of bandwidth  $w$ , or for short  $w$ -path, in a graph  $G$  is a sequence  $(v_1, \dots, v_n)$  of vertices such that for all  $v_i, v_j$  with  $1 \leq j - i \leq w$  the pair  $(v_i, v_j)$  is an edge in  $G$ , see Fig. 1 for an example of a 2-path. 1-paths are paths in the usual sense. It will be easy to show that for every  $w \geq 1$  the corresponding computational problem BANDWIDTH- $w$ -PATH, which asks for a given graph  $G$  and an integer  $k$  whether there exists a simple  $w$ -path of length  $k$  in  $G$ , is NP-complete.

The BANDWIDTH- $w$ -PATH problem for every  $w$  is fixed-parameter tractable in the parameter  $k$ , this will be shown according to the characterization of  $\text{FPT} \cap \text{NP}$  by Cai, Chen, Downey & Fellows [2] via an “FPT guess and check protocol” using witnesses of size only dependent on the parameter. The runtime obtained for our guess and check protocol, for the case  $w = 1$ , which is the LONGEST PATH problem, and seen as a deterministic exhaustive search algorithm, is worse than the algorithms of Monien [5] and Alon, Yuster, Zwick [1]. On the other hand, our algorithm is more easily stated and can immediately be applied to the BANDWIDTH- $w$ -PATH problem. Moreover, the algorithms of [5], [1] do not seem to give better FPT guess and check protocols.

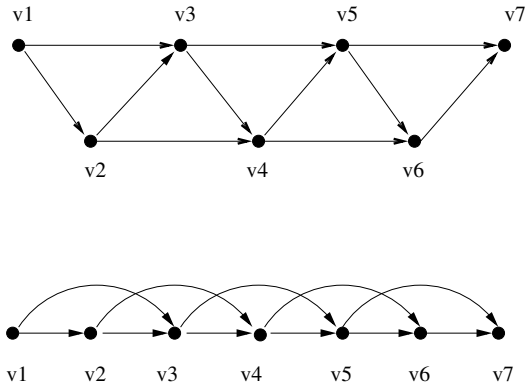


Fig. 1. Two drawings of the same 2-path of length 5, vertex-disjoint and deterministic

## 2 Paths of Constant Bandwidth

Let  $G$  be a digraph and let  $w, k \geq 1$ . A *path of bandwidth  $w$  and length  $k$*  in  $G$  is a sequence of  $k + w$  vertices  $(v_1, \dots, v_{k+w})$  such that the pair  $(v_i, v_{i+j})$  is an edge of  $G$  for every  $i$  with  $1 \leq i \leq k$  and every  $j$  with  $1 \leq j \leq w$ . A path of bandwidth  $w$  and length  $k$  will also be called  *$w$ -path of length  $k$*  or, even shorter,  *$(w, k)$ -path*. A 1-path of length  $k$  is a path of length  $k$  in the usual sense. (For a path of length  $k$  some authors count the number of vertices while others count the number of edges – which is one less. In this paper we count the number of edges.) In Figures 2 and 3 some 2-paths and 3-paths are shown. Note that a  $(w, 1)$ -path is a  $(w + 1)$ -clique: every two nodes are connected by an edge. A  $(w, k)$ -path can actually be seen as a sequence of  $k$   $(w + 1)$ -cliques with two subsequent cliques “glued” together by their common  $w$  elements.

A  $(w, k)$ -path  $(v_1, \dots, v_{k+w})$  is *vertex-disjoint* if all  $v_i$  are different from each other, it is *simple* if all  $k$   $w$ -tuples  $(v_1, \dots, v_w), (v_2, \dots, v_{w+1}), \dots, (v_k, \dots, v_{k+w})$  are different from each other. A vertex-disjoint  $(w, k)$ -path is simple, but not vice versa for  $w \geq 2$ , see Figure 3. A vertex-disjoint  $(w, k)$ -path, as a graph on its own, is the graph with  $k + w$  vertices having bandwidth  $w$  and a maximal set of edges, that is why we choose the name “bandwidth” for the number  $w$  (see [6], [4] for the definition of bandwidth of a graph).

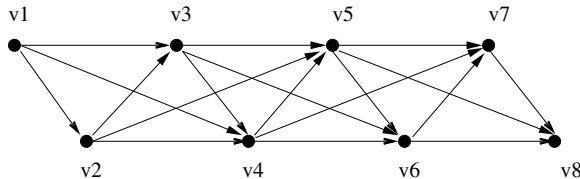
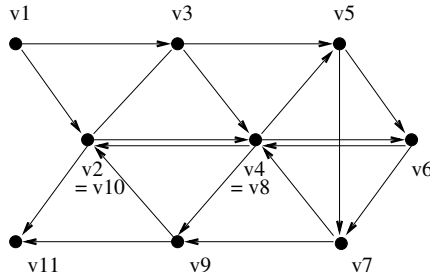


Fig. 2. A 3-path of length 5, vertex-disjoint and deterministic

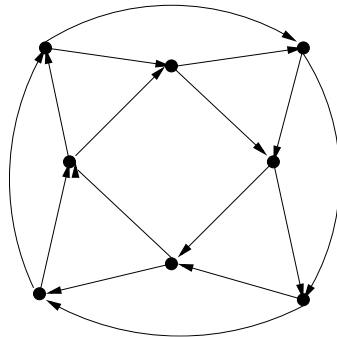


**Fig. 3.** A 2-path of length 10, deterministic and simple but not vertex-disjoint

Though the notion of  $w$ -paths within a graph  $G$  is a rather natural generalization of paths the authors could not find references for it in the literature. The closest concept found is the  $w$ -ray from Proskurowski & Telle [6], corresponding to a vertex-disjoint  $w$ -path (as a graph on its own).

A  $(w, k)$ -path  $(v_1, \dots, v_{k+w})$  is *deterministic* in  $G$  if for every  $1 \leq i \leq k$   $v_{i+w}$  is the only vertex in the graph  $G$  having the property that all edges  $(v_i, v_{i+w}), \dots, (v_{i+w-1}, v_{i+w})$  are edges of the graph. For example, a deterministic 1-path has the property that every vertex of it – besides the last one – has exactly one outgoing edge in  $G$ .

For  $w < k$ , a  $(w, k)$ -path  $(v_1, \dots, v_{k+w})$  is a *cycle of bandwidth  $w$  and length  $k$* , for short  $w$ -cycle of length  $k$  or  $(w, k)$ -cycle, if  $(v_{k+1}, \dots, v_{k+w}) = (v_1, \dots, v_w)$ . The cycle is *vertex-disjoint* if  $v_1, \dots, v_k$  are different from each other, it is *simple* if  $(v_1, \dots, v_{k+w-1})$  is a simple  $w$ -path, see Fig. 4 for an example. For undirected graphs the definitions can be transferred literally. For a fixed  $w$  let BANDWIDTH- $w$ -PATH be the set of pairs  $\langle G, k \rangle$  such that the digraph  $G$  contains a simple  $(w, k)$ -path. BANDWIDTH-1-PATH = LONGEST-PATH. Let BANDWIDTH-PATH be the double-parameterized problem consisting of the triples  $\langle G, w, k \rangle$  such that the digraph  $G$  contains a simple  $(w, k)$ -path.



**Fig. 4.** A 2-cycle of length 8, deterministic and vertex-disjoint

Some variations of these problems: Let the prefixes UNDIRECTED- and DISJOINT- in front of these problem names indicate that the input graph is undirected, or, independently, that the path to be found has to be not only simple but vertex-disjoint, respectively. Let CYCLE instead of PATH in a problem name denote that the path to be found has to be a cycle. Call these further 7 problems the *variations* of the BANDWIDTH-w-PATH problem, resp. BANDWIDTH-PATH.

**Proposition 1.** *For every  $w \geq 1$  the problem BANDWIDTH-w-PATH is NP-complete, likewise its variations.*

*Proof.* Obviously all problems are in NP. In order to show NP-completeness of BANDWIDTH-w-PATH we reduce LONGEST PATH to it. Let some directed graph  $G = (V, E)$  be given. Let the vertices  $V'$  of the graph  $\phi(G) = (V', E')$  consist of  $3w$  copies  $v_i$  with  $1 \leq i \leq 3w$  for each  $v \in V$  and  $w$  copies  $(u, v)_i$  with  $1 \leq i \leq w$  for each  $(u, v) \in E$  and let the edges be  $E' = \{(u_i, u_j) \mid i < j, u \in V\} \cup \{(u_{i+w}, (u, v)_j), ((u, v)_i, v_{j+w}) \mid 1 \leq j \leq i \leq w, (u, v) \in E\} \cup \{(u, v)_i, (u, v)_j \mid i < j, (u, v) \in E\}$ .

It holds:  $G$  has a simple path of length  $k$  iff  $\phi(G)$  has a simple  $w$ -path of length  $(2k + 3)w$  iff  $\phi(G)$  has a vertex-disjoint  $w$ -path of length  $(2k + 3)w$ . Observe that a  $w$ -path  $(\dots u_{w+1}, \dots, u_{2w}, (u, v)_1)$  in  $\phi(G)$  can only be continued by  $((u, v)_2, \dots, (u, v)_w, v_{w+1}, \dots, v_{2w}, \dots)$  which forces a simple  $w$ -path to be vertex-disjoint and to correspond with a path in  $G$ . Starting with  $(v_1, \dots, v_w, \dots)$  or ending with  $(\dots v_{2w+1}, \dots, v_{3w})$  allows to have the same length as starting in vertices corresponding to possibly unused edges in  $G$ . q.e.d.

DISJOINT-BANDWIDTH-PATH is NP-complete because LONGEST PATH is a subproblem and the length of a path is still at most linear. On the other hand, a simple path may have a length of  $\binom{n}{w}$  and we conjecture PSPACE-completeness for BANDWIDTH-PATH.

We mention that for fixed  $w$  the problem of searching for a *deterministic* simple  $w$ -path of a given length  $k$  can be done in PTIME by a straightforward marking algorithm.

### 3 Fixed-Parameter Tractability

The following notion is from Downey & Fellows [3] though it can already be found – without giving it a name – in Monien [5–p. 240, the two paragraphs before and after Th. 1, resp.].

**Definition 1 (fixed-parameter tractability [5], [3]).** *A computational problem consisting of pairs  $\langle x, k \rangle$  is fixed-parameter tractable in the parameter  $k$  if there is a deciding algorithm for it having run-time  $f(k) \cdot |x|^c$  for some recursive function  $f$  and some constant  $c$ .*

We use the following characterization of  $\text{FPT} \cap \text{NP}$  by Cai, Chen, Downey & Fellows [2]:

**Theorem 1 (Cai et al. [2]).** *A language  $L \in \text{NP}$  consisting of pairs  $\langle x, k \rangle$  is fixed-parameter tractable in the parameter  $k$  iff there exists a recursive function  $s(k)$  and a PTIME computable language  $C$  such that  $\langle x, k \rangle \in L \iff \exists y \leq s(k): \langle x, k, y \rangle \in C$ .*

We call the function  $s$  the *witness size function*, and the language  $C$  the *witness checker*, and we say that these two together form an FPT *guess and check protocol* for  $L$ .

**Theorem 2.** *For every  $w \geq 1$  the problem BANDWIDTH- $w$ -PATH is fixed parameter tractable in the parameter  $k$ , likewise its variations. More specifically, there exists an FPT guess and check protocol for it with a witness size function  $s(k) = \binom{k}{2} \cdot \log k$  and a witness checker having running time  $O(w \cdot k^2 \cdot |E|^w \cdot |V|^w)$ .*

*Proof.* We first consider the case  $w = 1$ , i.e. the LONGEST PATH problem. Afterwards we will see that the algorithm is generalizable to the BANDWIDTH- $w$ -PATH problem for  $w > 1$ . We state an FPT guess and check protocol for LONGEST PATH with the witness size function  $s(k) = \binom{k}{2} \cdot \log k$  and a witness checker with runtime  $O(k^2 \cdot |E| \cdot |V|)$ .

$a_{2,1}$	$a_{2,2}$	$a_{2,3}$	$a_{2,4}$	= $a_2$ =	1	1	0	0
$a_{3,1}$	$a_{3,2}$	$a_{2,3}$		= $a_3$ =	1	2	0	
$a_{4,1}$	$a_{4,2}$			= $a_4$ =	2	0		
$a_{5,1}$				= $a_5$ =	0			

**Fig. 5.** Witness table for a simple path of length 4

Let a digraph  $G$  with  $n$  vertices be given. We want to find out whether the graph contains a simple path  $p = (v_1, \dots, v_{k+1})$  of length  $k$ . We will work with *witnesses*. The intention of a witness is to tell the algorithm in the moment when it is trying to build an initial segment  $(v_1, \dots, v_i)$  of the simple path of length  $k$  which are the future vertices  $v_{i+1}, \dots, v_{k+1}$  of the simple path – so that the algorithm does not pick one of these future vertices as a part of the initial segment. Unfortunately, we cannot use the tuple  $(v_1, \dots, v_{k+1})$  as a witness, because that way we would have  $n^{k+1}$  potential witnesses, so we would need at least  $(k + 1) \log(n)$  bits to encode them, a number growing in  $n$ . But for the FPT guess and check protocol we need some witness size function  $s(k)$  only dependent on  $k$ .

We choose the following kind of witnesses. A *witness* for such a simple path of length  $k$  consists of  $k(k + 1)/2 = \binom{k+1}{2}$  numbers  $a_{i,j} \in \{0, 1, \dots, k\}$ , for  $2 \leq i \leq k + 1$  and  $j \in \{1, \dots, k - i + 2\}$ . The witness can be visualized as a half-matrix  $a$ , see Figure 5. Let  $a_i$  for  $2 \leq i \leq k + 1$  be the tuple  $(a_{i,1}, \dots, a_{i,k-i+2})$ .

We can restrict the witnesses to have these properties:  $a_i$  contains only numbers  $\leq i - 1$  and at least one 0. There is some redundancy, for example  $a_{k+1,1}$  will always be 0. Nevertheless, the order of magnitude of the witness size function  $s(k)$  does not seem to be improvable by these “little savings”.

For every witness  $a$  the main algorithm  $C$  does the following: In each of the  $k$  steps  $i = 2, 3, \dots, k + 1$  it uses  $a_i$  to compute for every vertex  $v$  a value  $f_{a,i}(v)$ , defined further below, which is either a vertex or has the value **nil** (standing for “not existing”), and stores this function for use in the following steps. The following pseudo code shows the main structure of the algorithm.

Main algorithm C

Input: graph  $G$ , number  $k \leq |G|$ , and a witness  $a$

for every vertex  $v$  set  $f_{a,1}(v) := v$ ;

for  $i = 2, \dots, k + 1$  do

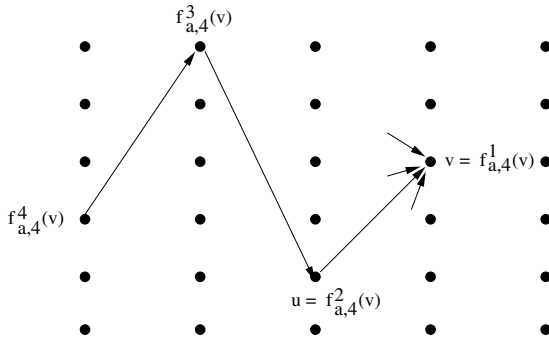
for every vertex  $v$  in  $G$  do

compute  $f_{a,i}(v)$  and store it;

if  $i = k + 1$  and  $f_{a,i}(v) \neq \mathbf{nil}$  ACCEPT and STOP;

REJECT and STOP;

The computation of the value  $f_{a,i}(v)$  – which is either **nil** or a vertex – is described in the pseudo code below. Assume w.l.o.g. that for each vertex there is a list of incoming edges (ending with the **nil** list element) in which the edges appear according to the order on the vertices. As a useful abbreviation let  $f_{a,i}^d(v)$  for a vertex  $v$  and  $d$  with  $1 \leq d \leq i + 1$  be defined via



**Fig. 6.** A “backward path”, starting in  $v$

$f_{a,i}^1(v) := v$ ,  $f_{a,i}^2(v) := f_{a,i}(v)$ , and  $f_{a,i}^{d+1}(v) := f_{a,i-1}^d(f_{a,i}(v))$  with this value being **nil** in case  $f_{a,i}(v)$  or  $f_{a,i-1}^d(f_{a,i}(v))$  equals **nil**. Intuitively,  $f_{a,i}^d(v)$  follows – starting in  $v$  – for growing  $d = 1, \dots, i + 1$  the “backward path” given by the  $f_{a,i-d}$ -functions, see Figure 6. The upper index  $d$  numbers the vertices of this path, and the witness elements  $a_{i,j} \geq 0$  will refer to this numbering. The information provided by  $a_{i,j} \geq 0$  means that the  $a_{i,j}$ -th vertex in the “backward path” starting with a vertex  $u$  is the “reason” to dismiss the  $j$ -th attempt to assign a possible predecessor  $u$  of  $v$  to  $f_{a,i}(v)$ . (The set  $F$  in the following pseudo code collects such “reasons”.)

By easy induction on  $i$ , the following invariant will be guaranteed for every witness  $a$ , every  $i$  with  $2 \leq i \leq k + 1$ , and every vertex  $v$ :

(Inv1) If  $f_{a,i}(v) \neq \mathbf{nil}$  then the “backward path”  $(f_{a,i}^i(v), \dots, f_{a,i}^2(v), f_{a,i}^1(v))$  is a simple path of length  $i - 1$ .

<p>Computing <math>f_{a,i}(v)</math></p> <p>Input: <math>i</math>, <math>a</math>, and <math>v</math>. Already computed: <math>f_{a,1}, \dots, f_{a,i-1}</math>.</p> <p>set <math>F := \{v\}</math>;</p> <p>set <math>j := 1</math>;</p> <p>if there are no incoming edges for <math>v</math> set <math>f_{a,i}(v) := \mathbf{nil}</math> and STOP;</p> <p>set <math>e = (u, v)</math> to be the first edge incoming to <math>v</math>;</p> <p>while <math>e \neq \mathbf{nil}</math> do</p> <p style="padding-left: 2em;">if <math>f_{a,i-1}(u) \neq \mathbf{nil}</math> and</p> <p style="padding-left: 4em;">none of the vertices <math>f_{a,i-1}^1(u), f_{a,i-1}^2(u), \dots, f_{a,i-1}^i(u)</math> is in <math>F</math> do</p> <p style="padding-left: 6em;">set <math>c := a_{i,j}</math>;</p> <p style="padding-left: 6em;">if <math>c = 0</math></p> <p style="padding-left: 8em;">set <math>f_{a,i}(v) := u</math> and STOP;</p> <p style="padding-left: 6em;">otherwise</p> <p style="padding-left: 8em;">set <math>F := F \cup \{f_{a,i-1}^c(u)\}</math>;</p> <p style="padding-left: 8em;">set <math>j := j + 1</math>;</p> <p style="padding-left: 4em;">set <math>e = (u, v) := \text{next edge going into } v</math>;</p> <p>set <math>f_{a,i}(v) := \mathbf{nil}</math> and STOP;</p>
---

Verification of the main algorithm  $C$ : If the algorithm accepts, then it has found for this witness  $a$  a vertex  $v$  such that  $f_{a,k+1}(v) \neq \mathbf{nil}$ . By invariant (Inv1), case  $i = k + 1$ , the backward path starting in  $v$  is a simple path of length  $k$ .

On the other hand assume that there is a simple path of length  $k$  in  $G$ . Let  $s = (s_1, \dots, s_{k+1})$  be the lexicographically smallest among them (largest weight on  $s_{k+1}$ , unlike, for example, with decimal numbers). With the knowledge of this path and its vertices we will construct a witness  $b$  such that the main algorithm will accept for witness  $b$ .

### Constructing $b$

Input:  $s_1, \dots, s_{k+1}$ .

for every vertex  $v$  set  $f_{b_1,1}(v) = v$ ;

for  $i = 2$  to  $k + 1$  do

    set  $e = (u, s_i) :=$  first edge going into  $s_i$ ;

    set  $F = \{s_i\}$ ;

    set  $j := 1$ ;

    repeat

        while  $f_{b_{i-1},i-1}(u) = \mathbf{nil}$

            or some of the vertices  $f_{b_{i-1},i-1}^1(u), \dots, f_{b_{i-1},i-1}^i(u)$  is in  $F$

                set  $e = (u, s_i) :=$  next edge going into  $s_i$ ;

        if there is a  $c \in \{1, \dots, i\}$  such that  $f_{b_{i-1},i-1}^c(u) \in \{s_{i+2}, \dots, s_{k+1}\}$

            set  $b_{i,j} := c$  for the smallest such  $c$ ;

            set  $F := F \cup \{f_{b_{i-1},i-1}^c(u)\}$ ;

            set  $j := j + 1$ ;

    until there is no such  $c$ ;

$b_{i,j} := 0$

    compute  $f_{b_i,i}(v)$  for all vertices  $v$ ;

The crucial invariant kept by this construction is the following:

(Inv2) For every  $i$  with  $2 \leq i \leq k + 1$  it holds:  $f_{b_i,i}(s_i) = s_{i-1}$ .

The invariant holds via induction on  $i$ : the construction of  $b_i$  prevents  $f_{b_i,i}(s_i)$  from choosing one of the vertices  $s_{i+1}, \dots, s_{k+1}$  which will be needed in the future but which would be – without the witness – unknown at step  $i$ . Because there are at most  $k - i + 1$  such vertices the repeat loop will always terminate and, moreover, the part  $b_i$  of the witness has sufficient size. For every  $2 \leq i \leq k + 1$  it is guaranteed that the computation of  $f_{b_i,i}(s_i)$  will terminate, i.e. will be not-**nil**, because at least  $(s_{i-1}, s_i)$  is a suitable edge, and this will be the first suitable edge

which  $f_{b,i}(s_i)$  will find, i.e.  $f_{b,i}(s_i) = s_{i-1}$ , because otherwise  $s = (s_1, \dots, s_{k+1})$  would not be lexicographically minimal. Invariant (Inv2) implies for  $i = k+1$  that the backward path  $(f_{b,k+1}^{k+1}(s_{k+1}), \dots, f_{b,k+1}^2(s_{k+1}), f_{b,k+1}^1(s_{k+1}))$  at  $s_{k+1}$  equals  $s = (s_1, \dots, s_{k+1})$ , i.e. the main algorithm  $C$  will accept the input graph for this witness  $b$  via a non-**nil** value of  $f_{b,k+1}$  at vertex  $s_{k+1}$ . This finishes the correctness proof for the FPT guess and check protocol.

The running time of all  $f_{a_i}(v)$  for a fixed  $i$  is  $O(k \cdot |E|)$  as it is dominated by checking the backward path of length  $\leq k$  for each edge incoming to  $v$  (we ignore some  $\log(k)$  factors for the comparison algorithms). Therefore, the main algorithm  $C$  has runtime  $O(k^2 \cdot |V| \cdot |E|)$ . Representing all witnesses can be done with  $\binom{k}{2} \cdot \log k$  bits, i.e. the witness size function can be chosen this way (note that the diagonal of the half matrix does not need to be stored – it can be assumed to consist of 0's). This finishes the proof that an FPT guess and check protocol exists for LONGEST PATH.

Cases  $w > 1$ . We first do a graph transformation. From the given graph  $G$  construct the following graph  $G'$ : Consider all  $w$ -tuples  $(v_1, \dots, v_w)$  of vertices of  $G$ . Make such a tuple a vertex of  $G'$  if the tuple represents a directed  $w$ -clique in  $G$ , i.e.  $(v_i, v_j)$  is an edge in  $G$  for  $1 \leq i < j \leq w$ . The edges in  $G'$  are defined to consist of the pairs of such  $w$ -cliques of the special form  $((v_1, \dots, v_w), (v_2, \dots, v_w, v_{w+1}))$  such that also  $(v_1, v_{w+1})$  is an edge in  $G$ . We have the property:  $G$  contains a simple  $w$ -path of length  $k$  iff  $G'$  contains a 1-path of length  $k$ . The witness checker consists therefore of this graph transformation and subsequently the checking algorithm  $C$  for  $w = 1$  running on  $G'$ . In total the checking takes  $O(w \cdot |V|^w \cdot |E|^w)$  time, the first  $w$  stems from a slightly higher comparison time for tuples. The witnesses size function does not change.

Variants: For the vertex disjoint case with  $w > 1$  it is not enough to do the graph transformation, one has to go inside the checking algorithm  $C$  and maintain the vertex lists appropriately. **q.e.d.**

## 4 Conclusions and Open Questions

We introduced for every  $w \geq 1$  the NP-complete problem BANDWIDTH- $w$ -PATH and showed that it is fixed-parameter tractable in the length parameter  $k$  by presenting an FPT guess and check protocol for it, according to the characterization of Cai et al. [2].

As an open problem we suggest to study whether the witness size function, especially for the case LONGEST PATH, can be improved from the quasi-quadratic function  $\binom{k}{2} \log k$  to some quasi-linear function, for example by the methods of Monien [5] or Alon, Yuster & Zwick [1].

## References

1. Alon, N., Yuster, R., Zwick, U.: Color-Coding. *J. ACM* **42** 4 (1995) 844–856
2. Cai, L., Chen, J., Downey, R.G., Fellows, M.R.: On the Structure of Parameterized Problems in NP. *Inf. Comput.* **123** 1 (1995) 38–49

3. Downey, R.G., Fellows, M.R.: Fixed-Parameter Intractability. Structure in Complexity Theory Conference (1992) 36–49
4. Garey, M.R., Johnson, D.S.: Computers and Intractability. Freeman, New York (1979)
5. Monien, B.: How to Find Long Paths Efficiently. Annals of Discrete Mathematics **25** (1985) 239–254
6. Proskurowski, A., Telle, J.A.: Classes of Graphs with Restricted Interval Models. Discrete Mathematics & Theoretical Computer Science **3** 4 (1999) 167–176