REPORTS IN INFORMATICS

ISSN 0333-3590

The Price of Connectedness in Expansions

Fedor V. Fomin, Pierre Fraigniaud, and Dimitrios M. Thilikos

REPORT NO 273 **May 2004**

Department of Informatics
UNIVERSITY OF BERGEN
Bergen, Norway

This report has URL http://www.ii.uib.no/publikasjoner/texrap/ps/2004-273.ps

Reports in Informatics from Department of Informatics, University of Bergen, Norway, is available at http://www.ii.uib.no/publikasjoner/texrap/.

Requests for paper copies of this report can be sent to:

Department of Informatics, University of Bergen, Høyteknologisenteret,
P.O. Box 7800, N-5020 Bergen, Norway

The Price of Connectedness in Expansions

Fedor V. Fomin
Dep. of Informatics
University of Bergen
fomin@ii.uib.no

Pierre Fraigniaud
CNRS
University of Paris Sud
pierre@lri.fr

Dimitrios M. Thilikos
Lenguatges i Sistemes Informàtics
UPC Barcelona
sedthilk@lsi.upc.es

6th May 2004

Abstract

Expansion is the way of generalizing different graph layout and searching problems. We initiate the study of connected expansion which naturally arises in a number of applications. Our main tool for this investigation is the branchwidth of a graph. In particular, we prove that any 2-edge-connected graph of branchwidth k has a *connected* branch decomposition of width k, i.e., a branch decomposition in which any cut separates two edge-sets that induce two connected subgraphs. Our proof is constructive, and is inspired from the existential proof of Seymour and Thomas (1994) for carvings. We also prove that the *connected* search number (i.e., connected pathwidth) of any n-node graph of branchwidth k is at most $O(k \log n)$ and this bound is the best possible for parameters k and n. A first consequence of these results is that, for any graph, the connected search number is at most $O(\log n)$ times larger than the (standard) search number. The only bound known so far held for trees only. Another consequence is that the connected search number can be approximated in polynomial time up to a factor $O(\log n \cdot \log OPT)$. That is, for any connected graph G, one can compute in polynomial time a connected search strategy for G that uses at most $O(\log n \cdot \log OPT)$ times the optimal number OPT of searchers. The ratio $O(\log n \cdot \log OPT)$ is the same as the best known approximation ratio for (standard) pathwidth, and any improvement for the approximation of connected search (i.e., connected pathwidth) would indeed produce an improvement for the approximation of pathwidth.

Keywords: Branchwidth, Graph Searching, Pathwidth, Expansion.

1 Introduction

Assume a set of inter-related functional units in a distributed system. The objective is to check the correctness of these units, one by one. In order to avoid checked units to be subject to the propagation of faults from neighboring unchecked units, the "checker" uses resources to protect checked units against unchecked neighboring ones. Once all neighboring units of a checked unit U has been checked, there is no need to protect U, and only the frontier between checked and unchecked units has to be guarded. The objective is to minimize the amount of resources required for the system to be entirely checked.

A closely related problem is the one consisting of a financial company $\mathcal{C}$ to acquiring market positions via the control of industrial sectors. Industrial sectors are inter-dependent. Assume three sectors S, S', and S'' owned by $\mathcal{C}$. If sector S depends (technologically, strategically, or financially) only on sectors S' and S'', then the control of these two latter sectors induces the control of sector S, in the sense that another company $\mathcal{C}'$ acquiring S would be subject to financial or technological decisions decided by $\mathcal{C}$. On the other hand, if a controlled sector S depends on a non-controlled sector S', then resources (shares, or budget) should be invested in S to avoid a strong dependence from S'. Perhaps the best would even be to acquire S'. The objective for the company is to minimize the amount of resources required to acquire a large fraction of the sectors, or even to acquire monopoly on all sectors.

Both problems can be modelled by an *expansion* problem in graphs. Let M be a set, and let α be a function mapping subsets of M to integers. A *k-expansion* in M is a sequence $\mathcal{A} = (A_0, \ldots, A_r)$ of subsets of M where:

- $A_0 = \emptyset$, and $A_r = M$;
- for each $i, 0 \leq i \leq r-1, |A_{i+1} - A_i| \leq 1$;
- for each $i, 0 \leq i \leq r-1, \alpha(A_i) \leq k$.

An expansion is called *monotone* if, additionally,

- for each $i, 0 \leq i \leq r-1, A_i \subseteq A_{i+1}$.

It was proven in [10] that for any k-expansion there exists a monotone k-expansion if α is a connectivity function, that is α is symmetric (i.e., for any $A \subseteq M, \alpha(A) = \alpha(M - A)$) and submodular (i.e., for any $A, B \subseteq M, \alpha(A \cap B) + \alpha(A \cup B)) \leq \alpha(A) + \alpha(B)$).

Given a graph $G = (V, E)$, a first example of an expansion is defined if for any $A \subseteq V, \alpha(A)$ is the set of edges with endpoints in both A and $M - A$. Then α is a connectivity function, and a graph G has a k-expansion if and only if its cutwidth is at most k. (Cutwidth is a standard parameter arising in VLSI, we refer to [14] for more information on this parameter.)

Another example of an expansion is defined if, for any $A \subseteq E$, $\alpha(A)$ is the number of vertices incident both to an edge in A and to an edge in $E - A$. Then α is again a connectivity function, and a graph G has a k-expansion if and only if its linear-width is at most k. (Linear-width [19] is a parameter closely related to pathwidth and search number – see also [18]).

Yet another natural example of expansion is the conquest game [10], where we have a set of countries subject to join some organization. At every moment of time we can either add a country to the union or expel an arbitrary number of countries. Adding countries and keeping them in the union needs some resources, say for guarding its border from the countries outside the union.

In all these examples it is a natural (and some times necessary) to demand to have *connected* expansions. For example, in conquest game the connectivity means that we can add a country to the union only if it is has a *common border* with some other country from the union. In search game it means that the cleared set should be connected which is necessary condition for agents to communicate to each other. Moreover, the same condition should be imposed in cases where the searchers cannot "jump" from one node to an non-adjacent one (e.g., cannot pass through the "walls" that determine the structure of the graph where the search takes place). The main question we are interested in this paper is:

Question: What additional price should we pay for the connected expansion?

In other words, what is the smallest l such that the existence of k-expansion yields the existence of a connected l-expansion? The proofs given in this extended abstract are stated in terms of the connectivity function α corresponding to linear-width. We stress that our results can also be stated in terms of general expansions. However instead of working with arbitrary finite sets M and functions α we decided to restrict ourselves to the special case when M is the edge set of a graph G and the function α is the number of separating vertices. One reason for such a decision is that this case is related to other important applications like graph searching and branchwidth. Another reason is that the proofs in this case become much more visible.

1.1 Expansion and Graph Searching

Let $G = (V, E)$ be a graph (with possible multiple edges and loops), and let $n = |V|$, and $m = |E|$. For subsets $A, B \subseteq E$, $A \cap B = \emptyset$, we define $\delta(A, B)$ to be the set of vertices adjacent to at least one edge in A and to at least one edge in B. We also use $\delta(A)$ to denote $\delta(A, E - A)$. A *k-expansion* in G is a sequence $X_0, X_1, \ldots, X_r$ where $X_i \subseteq E$ for every $i = 0, \ldots, r$, $X_0 = \emptyset$, $X_r = E$, and satisfying the following:

- $|X_{i+1} - X_i| \leq 1$ for every $i = 0, \ldots, r-1$;
- $|\delta(X_i)| \leq k$ for every $i = 0, \ldots, r$.

Using the terminology of [5], a k-expansion is hence a crusade of frontier at most k. The *expansion number* $\mathbf{x}(G)$ of G is the minimum k for which there is a k-expansion in G. A graph with expansion number k can thus be obtained by adding one edge after the other, whilst preserving that no more than k nodes are on the boundary between the current set of edges and the remaining edges. The expansion is *monotone* if

- $X_i \subseteq X_{i+1}$ for every $i = 0, \ldots, r-1$.

It is know [5] that, if G has a k-expansion, then it also has a monotone k-expansion. Moreover, according to [18], $\mathbf{x}(G)$ is equal to the linear-width of G, a parameter defined in [19]. The notion of expansion is strongly related to *graph searching*. A *search* strategy using k searchers in a graph G is a sequence of search steps consisting of one of the three following operations: (1) placing a searcher on a node; (2) moving a searcher along an edge; or (3) removing a searcher from a node. Operation 2 "clears" the edge traversed by the searcher. Originally, all edges are "contaminated", and the goal is to clear them all using the smallest number $\mathbf{s}(G)$ of searchers. A clear edge e is recontaminated if there exists a path between e and a contaminated edge, with no searcher on any node of the path. It is known [13] that recontamination does not help, in the sense that for any graph G there is a *monotone* search strategy using $\mathbf{s}(G)$ searchers, i.e., a search strategy using $\mathbf{s}(G)$ searchers and for which recontamination never occurs. Finding a search strategy in a graph using as few searchers as possible naturally arises when, e.g., the graph models a network penetrated by possibly harmful intruders, and the system sends agents to locate and block these intruders. Determining the search number of a graph is NP-hard [15], and the completeness follows from Lapaugh's theorem [13]. See also [11] for some different versions of graph searching.

The relation between expansion and search was noticed by [5, 18]. It follows that, for any graph G,

$$\mathbf{x}(G) \leq \mathbf{s}(G) \leq \mathbf{x}(G) + 1. \tag{1}$$

For instance $\mathbf{x}(P_n) = \mathbf{s}(P_n)$ and $\mathbf{x}(C_n) = \mathbf{s}(C_n)$ where P_n and C_n are repectively the path and the cycle of $n \geq 3$ nodes. For the $(n+1)$-node star $K_{1,n}$, we have $\mathbf{x}(K_{1,n}) = 1$, but $\mathbf{s}(K_{1,n}) = 2$ for $n \geq 3$. This relation between expansion and search has been used in [5] for the development of powerful techniques for proving the monotonicity of several variants of graph searching.

1.2 Branchwidth

Beside their practical applications, both expansions and search numbers found their main interest in their relation with the Graph Minor theory and the several related notions of *width* (e.g., pathwidth, treewidth, branchwidth, etc.). In particular, a *branch decomposition* [17] of a graph G is a tree T whose all internal nodes have degree 3, with a one-to-one correspondence between the leaves of T and the edges of G. Given an edge e of T, removing e from T results in two trees $T_1^{(e)}$ and $T_2^{(e)}$, and an *e-cut* is defined as the pair $\{E_1^{(e)}, E_2^{(e)}\}$, where $E_i^{(e)} \subset E$ is the set of leaves of $T_i^{(e)}$ for $i = 1, 2$. (Note that $E_1^{(e)} \cap E_2^{(e)} = \emptyset$ and $E_1^{(e)} \cup E_2^{(e)} = E$.) The width of T is defined as $\omega(T) = \max_e |\delta(E_1^{(e)})|$ where the maximum is taken over all e-cuts in T. The *branchwidth* $\mathbf{bw}(G)$ of G is then $\min_T \omega(T)$ where the minimum is taken over all branch decompositions T of G. Note that for 2-edge connected graphs $\mathbf{x}(G)$ can be alternatively defined as $\min_T \omega(T)$ where minimum is taken over all branch decompositions T of G with T being a caterpillar. It follows easily that for any graph G,

$$\mathbf{bw}(G) \leq \max\{\mathbf{x}(G), 2\}.$$

branchwidth was introduced by Robertson and Seymour in their Graph Minors series papers several years after treewidth. These parameters are rather close but many theorems in Graph Minors are easier to prove by using branchwidth instead of treewidth. Another powerful property of branchwidth is that it can be naturally generalized for hypergraphs and matroids. A good example of a generalization of Robertson and Seymour theory for matroids by using branchwidth is the recent paper by Geelen et al. [12]. A dual version of branchwidth, called *carving*, finds applications in the call routing problem in telephone networks [17]. As far as practical issues are concerned, branchwidth is related to the ability of recursively splitting the graph into several components separated by few nodes. In particular, there is an edge of the optimal branch decomposition of a graph G whose removal corresponds to splitting G into components of size at most $2m/3$ edges, and with at most $\mathbf{bw}(G)$ nodes in common.

For any graph G, we have:

$$\mathbf{bw}(G) - 1 \leq \mathbf{s}(G) = O(\mathbf{bw}(G) \cdot \log n). \tag{2}$$

This is because

- $\mathbf{pw}(G) \leq \mathbf{s}(G) \leq \mathbf{pw}(G) + 1$ where $\mathbf{pw}(G)$ is the pathwidth of G [4];
- $\mathbf{bw}(G) - 1 \leq \mathbf{tw}(G) \leq 3\,\mathbf{bw}(G)/2$ where $\mathbf{tw}(G)$ is the treewidth of G [16];
- $\mathbf{pw}(G) = O(\mathbf{tw}(G) \cdot \log n)$ (cf., e.g., [6]).

By (1) and (2), we have that $\mathbf{s}(G)$ and $\mathbf{x}(G)$ can be both approximated in polynomial time up to a factor $O(\log n \cdot \log OPT)$. Indeed, the branchwidth can be polynomially approximated up to a factor $\log OPT$ for any graph (cf. [1, 7]). Whether this $O(\log n \cdot \log OPT)$ factor can be improved has been an open problem for more than a decade, and it still holds despite several efforts for solving it.

1.3 Introducing Connectivity Constraints

This paper is concerned with connectivity issues that naturally arise when one asks the k-expansion to form a sequence of connected edge-induced subgraphs at every step, or the set of clear edges in a search strategy to be connected at every step. Similarly, splitting a graph into a small number of connected components of small size (say at most $2m/3$ edges) and connected through a small number of nodes leads to the notion of connected branchwidth in which the two sets of edges resulting from an e-cut are both connected (see [8] where the problem of balanced connected partition of graphs is analyzed).

More formally, a k-expansion $X_0, X_1, \ldots, X_r$ of a graph G is *connected* if, for any $i = 1, \ldots, r$, the subgraph induced by X_i is connected. The *connected expansion number* $\mathbf{cx}(G)$ of G is the minimum k for which there is a connected k-expansion in G. Similarly, a search strategy is *connected* if the set of clear edges induces a connected subgraph at every step of the search. The *connected search* number $\mathbf{cs}(G)$ of a graph G is the minimum k for which there is a connected search strategy in G using at most k searchers. Finally, a branch decomposition T of a graph G is *connected* if, for every e-cut in T, each of the resulting two sets of edges induces a connected subgraph of G.

1.4 Our Results

We present a polynomial-time algorithm that, given a branch decomposition T of a 2-edge-connected graph G of width k, returns a connected branch decomposition of G of width $\leq k$. Therefore, surprisingly, the connected branchwidth of any 2-edge-connected graph is equal to its branchwidth. In other words, *there is no additional price for imposing the connectedness in branch decompositions*. As a consequence, it is possible to partition any 2-edge-connected graph of branchwidth k into at most three connected subgraphs of size $\leq m/2$ edges, sharing at most k nodes. Our algorithm is partially inspired from the non-constructive proof of Seymour and Thomas [17] which shows that if there exists a *carving* of width $< k$ in a 2-connected graph G, then there exists a *bond* (i.e., connected) carving of width $< k$ in G. Our algorithm applies to branchwidth, and constructs the connected branch decomposition explicitly.

We show that the connected expansion cannot be too large in comparison with the expansion, and that the same holds for graph searching. More specifically, we first prove that

$$\mathbf{cx}(G) \leq \mathbf{bw}(G) \cdot (1 + \log_2 m) \tag{3}$$

for any connected graph G. Using the facts that $\mathbf{bw}(G) \leq \max\{\mathbf{x}(G), 2\}$, $\mathbf{x}(G) \leq \mathbf{s}(G)$ and $\mathbf{cs}(G) \leq \mathbf{cx}(G) + 1$ we obtain as a consequence that

$$\mathbf{cx}(G)/\mathbf{x}(G) \leq 1 + \log_2 m \text{ and } \mathbf{cs}(G)/\mathbf{s}(G) \leq 2 + \log_2 m$$

for any connected graph G. That is, the connected expansion is no more than an $O(\log n)$ factor away from the non-connected one, and the same holds for the search number. The bound in (3) is asymptotically optimal in the sense that there are trees T for which $\mathbf{cx}(T) = \Theta(\log n)$ (cf. [3]), and the branchwidth of any tree is at most 2.

Combining our polynomial-time algorithm for connected branchwidth with the established relation between expansion (resp., search) and connected expansion (resp., connected search) allows us to derive a polynomial-time $O(\log n \cdot \log OPT)$-approximation algorithm for connected expansion and connected search in arbitrary graphs. In the case of planar graphs, the approximation ratio can be reduced to $O(\log n)$ by using the fact that the branchwidth of a planar graph is computable

in polynomial time [17]. Reducing the approximation ratio $O(\log n \cdot \log OPT)$ for arbitrary graphs leads to the same type of difficulties as reducing the best known approximation ratio $O(\log n \log OPT)$ for standard search, i.e., pathwidth.

2 Branchwidth and Connected Branchwidth

In this section, we describe an algorithm called Make-it-Connected, satisfying the following:

Theorem 1 *In time $O(m^3)$, given a branch decomposition T of a 2-edge-connected graph G of width k, Algorithm Make-it-Connected returns a connected branch decomposition T' of G with width $\leq k$.*

Notation. Given a sub-tree A obtained from an e-cut of a branch decomposition S, we denote by $\delta(A)$ the set of nodes in G that belongs to $\delta(L)$ where L is the set of edges of G corresponding to the leaves of A. We also denote by $|A|$ the cardinality of L. For two disjoint sub-trees A and B of S, $\delta(A, B)$ is the set of nodes of G that have at least one incident edge that is a leaf of A, and at least one incident edge that is a leaf of B. Given a branch decomposition S, a *quartet* is an ordered set (A_1, A_2, B_1, B_2) of four mutually disjoint subtrees of S satisfying the following:

1. there is an edge $e = \{x, y\}$ of S such that the roots a_1 and a_2 of A_1 and A_2 are both adjacent to x in S, and the roots b_1 and b_2 of B_1 and B_2 are both adjacent to y in S; (cf. Figure 2.)
2. $\delta(A_1, B_1) \neq \emptyset$ and $\delta(A_2, B_2) \neq \emptyset$;
3. $\delta(A_1, A_2) = \emptyset$;

Notice that, by the above definition, the leaves corresponding to the subtrees A_1, A_2, B_1, B_2 form a 4-partition of E. Algorithm Make-it-Connected is described in Figure 1. It proceeds as follows. Given a quartet (A_1, A_2, B_1, B_2) in S, the algorithm replaces this quartet by (A_1, B_1, A_2, B_2), resulting in a tree S' obtained by connecting a_1 and b_1 to x, and a_2 and b_2 to y (See Fig. 2). Actually, if (A_1, A_2, B_1, B_2) and (A_1, A_2, B_2, B_1) are both quartets in S, then the algorithm considers the two possible replacements, and chooses the one that has smaller width. Clearly S' is also a branch decomposition of G. Note however that neither (A_1, B_1, A_2, B_2) nor (A_2, B_2, A_1, B_1) is a quartet in S' since $\delta(A_i, B_i) \neq \emptyset$ for $i = 1, 2$, by definition of the quartet (A_1, A_2, B_1, B_2) in S. Algorithm Make-it-Connected proceeds by successive replacements of quartets in the branch-decomposition. The algorithm stops when there is no quartet in the current branch-decomposition (we will later prove that such a situation eventually occurs). The proof of Theorem 1 proceeds through a sequence of lemmas.

Lemma 1 *The replacement of a quartet as specified in Algorithm Make-it-Connected does not increase the width of the branch-decomposition.*

Proof. The only possible change in the width can occur because of the cut separating $A_1 \cup A_2$ from $B_1 \cup B_2$. We consider two cases depending whether or not (A_1, A_2, B_2, B_1) is a quartet. If (A_1, A_2, B_2, B_1) is also a quartet (i.e., $\delta(A_1, B_2) \neq \emptyset$ and $\delta(A_2, B_1) \neq \emptyset$), then

$$|\delta(A_1 \cup B_1, A_2 \cup B_2)| + |\delta(A_1 \cup B_2, A_2 \cup B_1)| \leq |\delta(B_1)| + |\delta(B_2)|. \quad (4)$$

Indeed, if $u \in \delta(A_1 \cup B_1, A_2 \cup B_2) - \delta(B_1)$ then $u \in \delta(B_2)$ because $\delta(A_1, A_2) = \emptyset$. Similarly, if $u \in \delta(A_1 \cup B_2, A_2 \cup B_1) - \delta(B_2)$ then $u \in \delta(B_1)$. Therefore, any node

counted on the left hand side of Equation 4 appears as many time on the right hand side. Thus Equation 4 holds. Hence $|\delta(A_1 \cup B_1, A_2 \cup B_2)| + |\delta(A_1 \cup B_2, A_2 \cup B_1)| \leq 2k$, and thus the smallest of the two boundaries in certainly of size $\leq k$. If (A_1, A_2, B_2, B_1) is not a quartet, then assume, w.l.o.g., that $\delta(A_1, B_2) = \emptyset$. We get $|\delta(A_1 \cup B_1, A_2 \cup B_2)| \leq |\delta(B_1)| \leq k$. ■

Lemma 2 *The branch decomposition T' returned by Algorithm Make-it-Connected is connected.*

Proof. If not, then there exists an e-cut that splits T' into two subtrees A and B such that the leaves of A does not induce a connected subgraph in G. Actually, one can easily show that there exists such an e-cut so that A is the union of two disjoint subtrees A_1 and A_2 with $\delta(A_1, A_2) = \emptyset$. Among this latter type of e-cuts, we choose an e-cut such that $|A|$ is maximum for that property. The other subtree B of this e-cut contains at least two leaves since otherwise removing the single edge corresponding to the leaf of B would result in disconnecting the graph G, a contradiction with the fact that G is 2-edge-connected. We thus have B as the union of two disjoint subtrees B_1 and B_2 whose roots are adjacent to the root of B. Since $|A|$ is maximum, we have $\delta(A, B_i) \neq \emptyset$ for $i = 1, 2$. Moreover, since G is connected, and $\delta(A_1, A_2) = \emptyset$, we have $\delta(A_i, B) \neq \emptyset$ for $i = 1, 2$. Therefore either $\delta(A_1, B_1) \neq \emptyset$ and $\delta(A_2, B_2) \neq \emptyset$, or $\delta(A_1, B_2) \neq \emptyset$ and $\delta(A_2, B_1) \neq \emptyset$, or both. Hence, there is a quartet in T', a contradiction since, by construction, the tree returned by Algorithm Make-it-Connected has no quartet. ■

Lemma 3 *Algorithm Make-it-Connected terminates.*

Proof. We use a potential argument, based on a measure defined in [17] for carvings. Any internal node x of the branch-decomposition S is of degree 3, and thus it defines three subtrees S_1, S_2, S_3 whose roots are connected to x. Then let

$$\phi(S_1, S_2, S_3) = \begin{cases} 0 & \text{if } \delta(S_i, S_j) \neq \emptyset \text{ for any } i \neq j; \\ |S_\ell| - 1 & \text{if } \delta(S_i, S_j) = \emptyset \text{ for some } i \neq j, \text{ where } \ell \notin \{i, j\}. \end{cases}$$

This function is well defined because, since G is connected, if $\delta(S_i, S_j) = \emptyset$ for some $i \neq j$, then $\delta(S_i, S_\ell) \neq \emptyset$ and $\delta(S_j, S_\ell) \neq \emptyset$ for $\ell \in \{1, 2, 3\} - \{i, j\}$. Now, we define a potential function ϕ defined on any branch-decomposition S with set of internal node $I(S)$ by $\phi(S) = \sum_{I(S)} \phi(S_1, S_2, S_3)$. We show that, after any step of Algorithm Make-it-Connected, the potential ϕ strictly decreases. For that purpose, it is enough to prove that

$$\phi(A_1 \cup B_1, A_2, B_2) + \phi(A_1, B_1, A_2 \cup B_2) < \phi(A_1, A_2, B_1 \cup B_2) + \phi(A_1 \cup A_2, B_1, B_2)$$

for any quartet (A_1, A_2, B_1, B_2). First note that, since, by definition of a quartet, $\delta(A_1, A_2) = \emptyset$, we get that $\phi(A_1, A_2, B_1 \cup B_2) = |B_1| + |B_2| - 1$. Hence, for

$$L = \phi(A_1 \cup B_1, A_2, B_2) + \phi(A_1, B_1, A_2 \cup B_2) \text{ and } R = \phi(A_1 \cup A_2, B_1, B_2) + |B_1| + |B_2| - 1,$$

it is enough to prove that $L < R$ for any quartet (A_1, A_2, B_1, B_2). Since $R > 0$, the lemma is satisfied if $L = 0$. Thus we restrict our analysis to the case $L > 0$ which means that either $\phi(A_1 \cup B_1, A_2, B_2) > 0$ or $\phi(A_1, B_1, A_2 \cup B_2) > 0$. W.l.o.g. we will examine the case where $\phi(A_1, B_1, A_2 \cup B_2) > 0$ which excludes that $\delta(A_1, A_2 \cup B_2) \neq \emptyset$ and $\delta(B_1, A_2 \cup B_2) \neq \emptyset$ simultaneously hold. Hence we consider two cases:

Case 1: $\delta(A_1, A_2 \cup B_2) = \emptyset$. If $\delta(A_1 \cup B_1, A_2) \neq \emptyset$ and $\delta(A_1 \cup B_1, B_2) \neq \emptyset$ then $L = |B_1| - 1 < |B_1| + |B_2| - 1 \leq R$. Therefore, we consider two sub-cases:

- If $\delta(A_1 \cup B_1, A_2) = \emptyset$, then $L = |B_1| + |B_2| - 2 < |B_1| + |B_2| - 1 \leq R$.

- If $\delta(A_1 \cup B_1, B_2) = \emptyset$, then $\delta(B_1, B_2) = \emptyset$, and thus $R = |A_1| + |A_2| + |B_1| + |B_2| - 2$. It follows that $L < R$ because, by definition of a quartet, $\delta(A_i, B_i) \neq \emptyset$ for any $i = 1, 2$.

Case 2: $\delta(B_1, A_2 \cup B_2) = \emptyset$. Then $\delta(B_1, B_2) = \emptyset$, and thus $R = |A_1| + |A_2| + |B_1| + |B_2| - 2$. Hence, $L < R$ because $\delta(A_i, B_i) \neq \emptyset$ for any $i = 1, 2$.

In all cases, the inequality $L < R$ holds, which completes the proof. ■

Proof of Theorem 1. From Lemmas 1 and 2, if Algorithm Make-it-Connected terminates, then it returns a connected branch decomposition of width $\leq k$. From Lemma 3, the algorithm does terminates. To compute the execution time of the algorithm, let us consider the potential function ϕ defined in the proof of Lemma 3. Since $\phi(S_1, S_2, S_3) \leq m - 1$, we get that the potential cannot exceed $O(m^2)$. Thus there are $O(m^2)$ updates of the branch-decomposition. Each update is local to the subtree of six nodes interconnecting the roots a_1, a_2, b_1, b_2 of A_1, A_2, B_1, B_2. For each of the edges $\{x, a_1\}, \{x, b_1\}, \{y, a_2\}, \{y, b_2\}$, deciding whether the edge defines a quartet takes $O(m)$ times. Thus Algorithm Make-it-Connected completes in $O(m^3)$ times. ■

3 Connected Expansion and Connected Search

We first show the following bound on the connected expansion. A direct consequence of this bound is that, for any connected graph G, $\mathbf{cx}(G) \leq \mathbf{bw}(G) \cdot (1 + \log_2 m)$.

Theorem 2 *Given a branch decomposition T of width k for a connected graph G, one can compute in $O(m^3)$-time a connected $k(1 + \log_2 m)$-expansion $X_0, X_1, \ldots, X_m$ in G. This result holds even if the first edge X_1 of the expansion is fixed.*

To prove Theorem 2, we use the following lemma which assume given a *connected* branch decomposition.

Lemma 4 *Given a connected branch decomposition T of width k for a graph G, and given any edge e of G, one can compute in $O(m^3)$-time a connected $(k \log_2 m)$-expansion $X_0, X_1, \ldots, X_m$ in G with $X_1 = \{e\}$.*

Proof. The proof is by induction on the number m of edges in G. For any $m \geq 1$, let $\mathcal{P}_m$ be the following property: for any $k \geq 0$, given a connected branch decomposition T of width k for a graph G with m edges, and given any edge e of G, one can compute in $O(m^3)$-time a connected $(k \log_2 m)$-expansion $X_0, X_1, \ldots, X_m$ in G with $X_1 = \{e\}$.

If $m = 1$, then there is a connected 0-expansion in G, and thus $\mathcal{P}_1$ holds. If $m = 2$, then G is either a path of three nodes, or two nodes linked by an edge and a loop around one of the two nodes, or two loops around a node, or two nodes joined by a double edge. In the first three cases, $\mathbf{bw}(G) = 1$. In the latter case $\mathbf{bw}(G) = 2$. In all cases though, one can construct a connected $\mathbf{bw}(G)$-expansion X_0, X_1, X_2 in G starting from any edge. Thus $\mathcal{P}_2$ holds.

Let $m > 2$. Assume that $\mathcal{P}_q$ holds for every $2 \leq q \leq m - 1$, and consider $\mathcal{P}_m$. There exists a node x of T whose removal results in three disjoint subtrees T_1, T_2, and T_3, with $|T_i| \leq \lfloor m/2 \rfloor$ for any $i \in \{1, 2, 3\}$ (recall that $|T_i|$ denotes the number of leaves of T_i). Since T is a connected branch decomposition, the leaves of each of these subtrees induces three connected subgraphs G_1, G_2, G_3 of G, and, for any $i \in \{1, 2, 3\}$, T_i is a connected branch decomposition of G_i. Given a set of edges X in $G_i = (V_i, E_i)$, we denote by $\delta_{G_i}(X)$ the set of nodes of G_i that has at least one incident edge in X, and at least one incident edge in $E_i - X$.

Since $m > 2$, $\lfloor m/2 \rfloor < m-1$, and thus each G_i (with $m_i \leq \lfloor m/2 \rfloor$ edges) satisfies the induction hypothesis. So let e be an edge of G. This edge is an edge of some G_i. Assume, w.l.o.g., that e is an edge of G_1. By induction, let us consider the $(k \log_2 m_1)$-expansion $X_0, X_1, \ldots, X_{m_1}$ in G_1 with $X_1 = \{e\}$. Removing the edge connecting x to the root of T_3 in T results in a connected subgraph $G_1 \cup G_2$. Thus, there is a node u in G that has at least one incident edge in G_1, and at least one incident edge f in G_2. By induction, let us consider the $(k \log_2 m_2)$-expansion $Y_0, Y_1, \ldots, Y_{m_2}$ in G_2 with $Y_1 = \{f\}$. Finally, since G is connected, and $G = G_1 \cup G_2 \cup G_3$, there is a node v in G that has at least one incident edge in $G_1 \cup G_2$, and at least one incident edge g in G_3. By induction, let us consider the $(k \log_2 m_3)$-expansion $Z_0, Z_1, \ldots, Z_{m_3}$ in G_3 with $Z_1 = \{g\}$. We obtain the expansion

$$X_0, X_1, \ldots, X_{m_1}, X_{m_1} \cup Y_1, \ldots, X_{m_1} \cup Y_{m_2}, X_{m_1} \cup Y_{m_2} \cup Z_1, \ldots, X_{m_1} \cup Y_{m_2} \cup Z_{m_3}$$

in G. It remains to bound the frontier of this expansion. We have

$$\delta(X_i) \leq \delta_{G_1}(X_i) + k \leq k \log_2 m_1 + k \leq k \log_2 \lfloor m/2 \rfloor + k \leq k \log_2 m/2 + k \leq k \log_2 m.$$

We also have

$$\delta(X_{m_1} \cup Y_i) \leq \delta_{G_2}(Y_i) + \delta(X_{m_1} \cup Y_i, G_3) \leq k \log_2 \lfloor m/2 \rfloor + k \leq k \log_2 m.$$

Finally, we have

$$\delta(X_{m_1} \cup Y_{m_2} \cup Z_i) \leq \delta_{G_3}(Z_i) + \delta(Z_i, G_1 \cup G_2) \leq k \log_2 \lfloor m/2 \rfloor + k \leq k \log_2 m.$$

Hence $\mathcal{P}_m$ is satisfied.

To complete the proof of the lemma, we observe that the time $\tau(m)$ needed to construct the expansion in an m-edge graph satisfies $\tau(m) \leq 3\tau(m/2)$, and thus the complexity of the construction is $3^{\log_2 m} = O(m^3)$. ■

Proof of Theorem 2. If G is 2-edge-connected, then, by application of Theorem 1, one can compute in $O(m^3)$ time a connected branch decomposition T' of G of width $\leq k$. The requested expansion is then obtained by application of Lemma 4.

If G is not 2-edge-connected, then we add a double edge to each isthmus (i.e., cut-edge) in G so that the resulting graph G' is 2-edge-connected. We obtain $\mathbf{bw}(G') \leq \max\{2, \mathbf{bw}(G)\}$. More precisely, given the branch decomposition T of G, one can construct a branch decomposition T' of G' such that $\omega(T') \leq \max\{2, \omega(T)\}$. Hence, we treat $k \leq 1$ as a special case.

If $k = 0$, then $\mathbf{bw}(G) = 0$ and thus G consists of a single edge. Therefore $\mathbf{cx}(G) = 0$, and Theorem 2 holds. If $k = 1$, then $\mathbf{bw}(G) \leq 1$ and thus G is a star $K_{1,n-1}$. If G has one edge, then Theorem 2 holds from the analysis of the case $k = 0$. Finally, if $m \geq 2$, then $\mathbf{cx}(G) = 1 \leq \log_2 m$ and Theorem 2 holds.

Assume now that $k \geq 2$. We construct G' and T', and then compute a connected branch decomposition T'' of G' in time $O(m^3)$. We have $\omega(T'') \leq \omega(T') \leq \omega(T) = k$. By application of Lemma 4, we obtain a connected $k \log m'$-expansion in G'. By removing in this expansion the second occurrence of every double edge added to isthmuses, we obtain a connected $k \log_2 m'$-expansion in G. We complete the proof by noticing that $m' \leq 2m$. ■

Corollary 1 *For any graph G, $\mathbf{cx}(G)/\mathbf{x}(G \leq 1 + \log_2 m$ and $\mathbf{cs}(G)/\mathbf{s}(G) \leq 2 + \log_2 m$.*

Proof. Theorem 2 implies that $\mathbf{cx}(G) \leq \mathbf{bw}(G) \cdot (1 + \log_2 m)$. Hence the corollary follows from the simple facts that $\mathbf{bw}(G) \leq \mathbf{x}(G) \leq \mathbf{s}(G)$, and $\mathbf{cs}(G) \leq \mathbf{cx}(G) + 1$. ■

4 Approximation Algorithms

Let $t(m)$ be the time-complexity of the fastest algorithm for approximating the branchwidth of a graph, up to a factor $O(\log OPT)$. Combining the results of the previous sections, we get:

Theorem 3 *There exists an $O(t(m)+m^3)$-time $O(\log n \log \mathbf{bw}(G))$-approximation algorithm for the connected expansion and for the connected search. More precisely, given any connected graph G, the algorithm returns an $O(\mathbf{cx}(G)\ \log n \log \mathbf{cx}(G))$-expansion in G, as well as a connected search strategy for G using at most $O(\mathbf{cs}(G) \log n \log \mathbf{cs}(G))$ searchers.*

Proof. In $t(m)$ time, we compute a branch-decomposition T for the input graph G such that $\omega(T) \leq \mathbf{bw}(G) \cdot O(\log \mathbf{bw}(G))$. If G is not 2-edge-connected, then we add a double edge to every bridge to obtain G' that is 2-edge-connected, and we modify the branch decomposition accordingly, resulting in T'. In $O(m^3)$ time, we use Algorithm Make-it-Connected (cf. Theorem 1), taking as input the branch-decomposition T'. It returns a connected branch decomposition T'' of width $\leq \mathbf{bw}(G) \cdot O(\log \mathbf{bw}(G))$ for G'. Finally, we apply the construction in the proof of Theorem 2 to get a connected $(\mathbf{bw}(G) \cdot O(\log \mathbf{bw}(G)) \cdot O(\log m))$-expansion in G. Since $\mathbf{x}(G) = \Omega(\mathbf{bw}(G))$, the result follows for the expansion. The same holds for search too since a connected k-expansion in G can be trivially transformed into a connected search in G using at most $k+1$ searchers. ■

Remark. The branchwidth of planar graphs can be computed exactly, in polynomial time (cf. [17]). Therefore, the approximation ratio for connected search and connected expansion can be reduced to $O(\log n)$ in planar graphs. Finally, according to the results in [9], the same holds for more general graph classes like the K_5-minor free graphs or the $K_{3,3}$-minor free graphs.

5 Conclusion

In this paper, we have described an algorithm that transforms a branch decomposition of a 2-edge-connected graph into a connected one, without increasing the width of the branch decomposition. On the other hand, we have also described an algorithm that constructs a connected expansion (and a connected search strategy) in a graph from any connected branch decomposition of this graph. Combining these two results we were able to derive bounds on the "price of connectedness". In particular, we have shown that $\mathbf{cx}(G)/\mathbf{x}(G) \leq 1 + \log |E(G)|$ for any connected graph G. Moreover, our algorithms, combined with previously know algorithms for approximating the branchwidth of a graph, allow us to derive $O(\log n \log OPT)$-approximation algorithm for the connected expansion and for the connected search. Our paper rises several important questions.

- A first direction for further researches would consist to generalize Theorem 1. For instance, is it true that, for any $k \geq 1$, there exists a constant $f(k)$ such that if G is $f(k)$-edge-connected then there exists a k-connected branch decomposition of G of width $\mathbf{bw}(G)$?
- We have addressed monotonous connected search strategies, and monotonous connected expansions. Under the connectivity constraint, it is however not known whether recontamination helps or not. Solving this question appears to be a challenging task because the connectivity constraint disables us from using most existing tools for monotonicity proofs (cf, e.g., [5, 10]). The only

class of graphs for which it is known that recontamination does not help for connected search is the class of trees [2]. More generally, does the existence of connected k-expansion in a set M always implies the existence of a monotone connected k-expansion?

- Finally, we noticed that the bound $\mathbf{cx}(G) \leq \mathbf{bw}(G) \cdot (1 + \log_2 m)$ is asymptotically tight. We are however concerned with the tightness of the bounds $\mathbf{cs}/\mathbf{s}$ and $\mathbf{cx}/\mathbf{x}$ in Corollary 1. It is known that these ratios can be decreased to 2 in the specific case of trees [3]. Does this latter bound hold for any graph? Does there exist graphs G for which $\mathbf{cx}(G)/\mathbf{x}(G) = \Omega(\log n)$?

Acknowledgments: All authors are thankful to Ioan Todinca for fruitful discussions and comments. F. Fomin is supported by Norges forskningsråd projects 160233/V30 and 160778/V30. P. Fraigniaud is supported by the project "PairA-Pair" of the ACI Masses de Données. D. Thilikos is supported by the EU within the 6th Framework Programme under contract 001907 (DELIS) and by the Spanish CICYT project TIC-2002-04498-C05-03 (TRACER).

References

[1] E. Amir. Efficient approximation for triangulation of minimum treewidth. In 17th Conference on Uncertainty in Artificial Intelligence (UAI), 2001.

[2] L. Barrière, P. Flocchini, P. Fraigniaud, and N. Santoro. Capture of an intruder by mobile agents. In 14th ACM Symp. on Parallel Algorithms and Architectures (SPAA), pages 200-209, 2002.

[3] L. Barrière, P. Fraigniaud, N. Santoro, and D. Thilikos. Searching is not jumping. In 29th Workshop on Graph Theoretic Concepts in Computer Science (WG), Springer-Verlag, LNCS 2880, pages 34–45, 2003.

[4] D. Bienstock, Graph searching, path-width, tree-width and related problems (a survey), DIMACS Ser. in Discrete Mathematics and Theoretical Computer Science, 5 (1991), pp. 33–49.

[5] D. Bienstock and P. Seymour. Monotonicity in graph searching. Journal of Algorithms 12:239–245, 1991.

[6] H. L. Bodlaender, A partial k-arboretum of graphs with bounded treewidth. Theor. Comp. Sc. 209:1–45, 1998.

[7] V. Bouchitté, D. Kratsch, H. Müller, and I. Todinca. On treewidth approximations. Discrete Applied Mathematics 136:183-196, 2004.

[8] J. Chlebíková. Approximating the maximally balanced connected partition problem in graphs. Information Processing Letters 60, page 225-230, 1996.

[9] E. Demaine, M. Hajiaghayi, and D. Thilikos. 1.5-Approximation for Treewidth of Graphs Excluding a Graph with One Crossing as a Minor. In 5th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2002), Lecture Notes in Computer Science, volume 2462, Rome, Italy, pages 67-80, 2002.

[10] F. Fomin and D. Thilikos. On the monotonicity of games generated by symmetric submodular functions. Discrete Applied Mathematics 131(2):323-335, 2003.

[11] M. Franklin, Z. Galil, and M. Yung, Eavesdropping games: A graph-theoretic approach to privacy in distributed systems, Journal of the ACM, 47:225–243, 2000.

[12] J. F. Geelen, A. M. H. Gerards, and G. Whittle, branchwidth and well-quasi-ordering in matroids and graphs, J. Combin. Theory Ser. B 84:270–290, 2002.

[13] A. Lapaugh. Recontamination does not help to search a graph. Journal of the ACM 40(2):224–245, 1993.

[14] F. S. Makedon and I. H. Sudborough, On minimizing width in linear layouts, Discrete Appl. Math., 23:243–265, 1989.

[15] N. Megiddo, S. Hakimi, M. Garey, D. Johnson and C. Papadimitriou. The complexity of searching a graph. Journal of the ACM 35(1):18–44, 1988.

[16] N. Robertson and P. D. Seymour, Graph minors. X. Obstructions to tree-decomposition, J. Combin. Theory Ser. B, 52:153–190, 1991.

[17] P. Seymour and R. Thomas. Call routing and the rat-catcher. Combinatorica 14(2):217–241, 1994.

[18] D. Thilikos. Algorithms and obstructions for linear-width and related search parameters. *Discrete Appl. Math.*, 105(1-3):239–271, 2000.

[19] R. Thomas, Tree decompositions of graphs. Lecture notes, 1996. Georgia Institut of Technology, Atlanta, Georgia, 30332, USA.

```
input: branch decomposition T of a 2-edge-connected graph of width k ;
output: connected branch decomposition T' of width ≤ k;
begin
    S := T;
    While there exists a quartet (A1, A2, B1, B2) in S do
       replace (A1, A2, B1, B2) in S by (A1, B1, A2, B2) to get S';
       if (A1, A2, B2, B1) is not a quartet in S then S := S'
       else
          replace (A1, A2, B1, B2) in S by (A1, B2, A2, B1) to get S'';
          if ω(S') ≤ ω(S'') then S := S' else S := S'';
       endif
    endwhile
    T' := S;
end
```

Figure 1: Algorithm Make-it-Connected

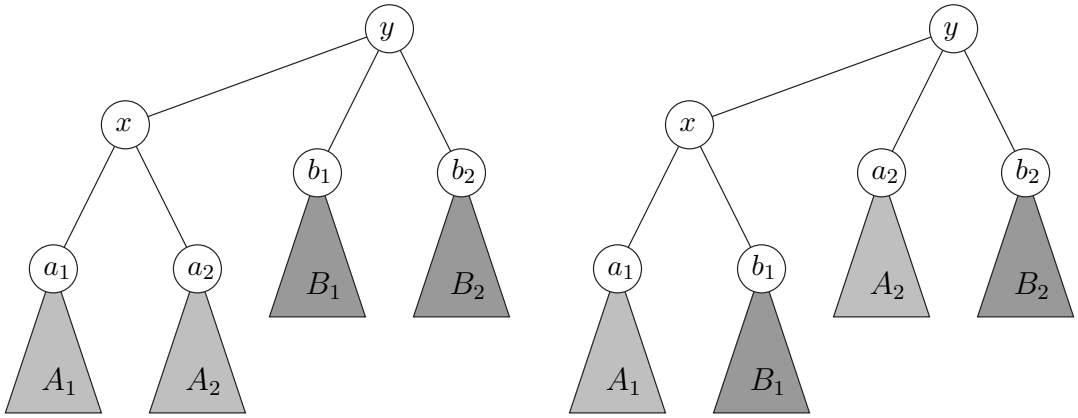

Figure 2: Replacing quartets in Make-it-Connected