Also available at http://amc.imfm.si
ISSN 1855-3966 (printed edn.), ISSN 1855-3974 (electronic edn.)
ARS MATHEMATICA CONTEMPORANEA 2 (2009) 191–205

On Cartesian skeletons of graphs

Richard H. Hammack
Department of Mathematics and Applied Mathematics
Virginia Commonwealth University, Richmond, VA, USA

Wilfried Imrich
Chair of Applied Mathematics, Montanuniversität Leoben
A-8700 Leoben, Austria

Received 27 July 2009, accepted 9 September 2009, published online 30 September 2009

Abstract

Under suitable conditions of connectivity or non-bipartiteness, each of the three standard graph products (the Cartesian product, the direct product and the strong product) satisfies the unique prime factorization property, and there are polynomial algorithms to determine the prime factors. This is most easily proved for the Cartesian product. For the other products, current proofs involve a notion of a *Cartesian skeleton* which transfers their multiplication properties to the Cartesian product.

The present article introduces simplified definitions of Cartesian skeletons for the direct and strong products, and provides new, fast and transparent algorithms for their construction. Since the complexity of the prime factorization of the direct and the strong product is determined by the complexity of the construction of the Cartesian skeleton, the new algorithms also improve the complexity of the prime factorizations of graphs with respect to the direct and the strong product.

We indicate how these simplifications fit into the existing literature.

Keywords: Graph product, Cartesian skeleton, prime factorization of graphs, graph algorithms.

Math. Subj. Class.: 05C85, 05C99

1 Introduction

We consider finite graphs $G = (V(G), E(G))$ which may have loops but not multiple edges. We let Γ_0 denote the class of all such graphs, while $\Gamma \subset \Gamma_0$ is the class of graphs without loops. An edge joining g to g' is denoted gg'.

E-mail addresses: rhammack@vcu.edu (Richard H. Hammack), imrich@unileoben.ac.at (Wilfried Imrich)

We assume our reader is quite familiar with the three standard graph products. Nonetheless, as the notation and lexicon are not universal, we collect the definitions and some main ideas in this introduction. Given graphs H and K, one can form the Cartesian product $H\Box K$, the direct product $H \times K$ and the strong product $H \boxtimes K$. Each has as a vertex set the Cartesian product $V(H) \times V(K)$ of sets. The edge sets are as follows.

$$\begin{aligned} E(H\Box K) &= \{(h,k)(h',k') : hh' \in E(H), k = k', \text{ or } h = h', kk' \in E(K)\} \\ E(H \times K) &= \{(h,k)(h',k') : hh' \in E(H) \text{ and } kk' \in E(K)\} \\ E(H \boxtimes K) &= E(H\Box K) \cup E(H \times K) \end{aligned}$$

These products are significant in that they are the only nontrivial associative and commutative products whose projections onto the factors are weak homomorphisims. (See Appendix C of [7].) Figure 1 shows specific examples.

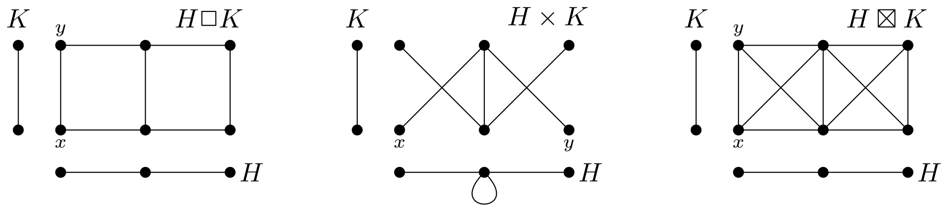


Figure 1: The three standard graph products

The one-vertex complete graph K_1 serves as a unit for $\Box$ and $\boxtimes$, as $K_1\Box H \cong H$ and $K_1 \boxtimes H \cong H$ for all graphs H. The graph K_1^s consisting of a single vertex on which there is a loop satisfies $K_1^s \times H \cong H$, so it is the unit for $\times$. A graph is called *prime* with respect to a particular product if whenever it is expressed as a product of two factors, one of the factors is the unit for the product.

Sabidussi [12] and Vizing [13] first proved that every connected graph in Γ has a unique factorization over the Cartesian product into prime factors in Γ. (See [8] for a more recent approach.) Moreover, efficient algorithms exist [4, 9] for determining the prime factors.

Any connected graph in Γ also has a unique prime factorization over the strong product, a result that was first proved in [3]. Subsequently, this same result was implied by McKenzie's general work [10] on products of relational structures. McKenzie's results also imply that the class of connected non-bipartite graphs in Γ_0 obey unique prime factorization over the direct product.

Articles [6, 5] present polynomial algorithms that determine the prime factorizations of graphs over the direct and strong products, and both approaches are adapted in [7]. These approaches involve construction of a so-called "Cartesian Skeleton," a graph that envelops a given graph G in such a way that factorizations of G over $\times$ (or $\boxtimes$) induce factorizations of the Cartesian skeleton over $\Box$. This links prime factorizations over $\times$ (or $\boxtimes$) to factorizations over $\Box$, and algorithms for prime factorization over $\Box$ are then applied to the Cartesian skeleton and transferred back to factorizations of G over $\times$ (or $\boxtimes$).

Cartesian skeletons are defined algorithmically in [6] and [5]. Unfortunately, the algorithmic approach makes it difficult to deduce even simple properties of skeletons and does not even ensure uniqueness. The purpose of the current paper is to provide simple non-algorithmic definitions of skeletons, and to describe separate algorithms for their construction. Our definitions clarify the structure of skeletons and consequently serve the dual purpose of simplifying the approaches of [6, 5] and potentially leading to new results and insights.

Our article is organized as follows. The rest of the introduction collects some necessary preliminary results. In Section 2 we define the Cartesian skeleton $S(G)$ of a graph G, and show that it satisfies $S(H \times K) = S(H) \Box S(K)$ for so-called "R-thin" graphs. We also show that $S(G)$ is connected provided that G is connected and non-bipartite. We briefly indicate how the skeleton fits into existing results that prove unique prime factorization over $\times$, and we describe algorithms for its construction. Section 3 introduces a modification of $S(G)$, which we call $S[G]$. We show $S[H \boxtimes K] = S[H] \Box S[K]$, and we obtain analogues of the results in Section 2. The last section pertains to the implications of the new algorithms for the prime factorization of graphs.

Our discussion is phrased in terms of vertex neighborhoods. The *(open) neighborhood* of a vertex $g \in V(G)$ is the set $N_G(g) = \{g' : gg' \in E(G)\}$. (If there is a loop at g, then $g \in N_G(g)$.) The *closed neighborhood* of g is $N_G[g] = N_G(g) \cup \{g\}$. The *degree* of g is $\deg(g) = |N_G(g)|$. It is an easy (and useful) fact that $N_{H\times K}(h, k) = N_H(h) \times N_K(k)$ and $N_{H\boxtimes K}[(h, k)] = N_H[h] \times N_K[k]$.

The notion of *thinness* is another important idea. To motivate this, we refer to Figure 1. Notice that the only automorphism of $H \Box K$ that interchanges vertices x and y is induced by the nontrivial automorphism of the factor K. In fact, it is a general property [7, Corollary 4.16] that any automorphism of a Cartesian product is generated by automorphisms of its prime factors (and transpositions of isomorphic factors). By contrast, $H \times K$ (in Figure 1) admits an automorphism that interchanges x and y and fixes all other vertices, and this automorphism is not induced by any automorphism of the factors. It is precisely the condition $N_{H\times K}(x) = N_{H\times K}(y)$ that permits the interchange of x and y. Similarly, in $H \boxtimes K$ we have $N_{H\boxtimes K}[x] = N_{H\boxtimes K}[y]$, and this permits an automorphism of $H \boxtimes K$ that interchanges x and y and is constant on all other vertices. This lack of rigidity tends to complicate discussion of prime factorizations over $\times$ and $\boxtimes$. Consequently we define a graph G to be *R-thin* if $N_G(x) = N_G(y)$ implies $x = y$ for all $x, y \in V(G)$. Likewise G is *S-thin* if $N_G[x] = N_G[y]$ implies $x = y$ for all $x, y \in V(G)$. In discussions of prime factorizations over $\times$ (respectively $\boxtimes$) it is often helpful to impose the condition of R-thinness (respectively S-thinness).

Recall [7, Corollaries 5.31 and 5.11] that $H \times K$ (respectively $H \boxtimes K$) is R-thin (respectively S-thin) if and only if both H and K are R-thin (respectively S-thin).

2 The Cartesian Skeleton for the Direct Product

In this section we define the Cartesian skeleton $S(G)$ of an arbitrary graph G in Γ_0 as a certain graph having the same vertex set as G. We prove that $S(G)$ is connected provided G is connected and non-bipartite, and show $S(H \times K) = S(H) \Box S(K)$ for R-thin graphs. Our point of departure is the Boolean square.

Boolean Squares

The *Boolean square* of a graph G is the graph G^s with $V(G^s) = V(G)$ and $E(G^s) = \{xy : N_G(x) \cap N_G(y) \neq \emptyset\}$. Thus, xy is an edge of G^s if and only if G has an x-y walk of length two. For example, K_n^s is K_n with a loop added to each vertex. We note in passing that the adjacency matrix of G^s is the Boolean square of the adjacency matrix of G, that is, if G has adjacency matrix A then the matrix of G^s is obtained from A^2 by replacing each nonzero entry by 1.

Observe that if G has an x-y walk W of even length, then G^s has an x-y walk of length $|W|/2$ on alternate vertices of W. It follows that G^s is connected if G is connected and has an odd cycle. (For the presence of an odd cycle guarantees an even walk between any two vertices of G.) On the other hand if G is connected and bipartite, then G^s has exactly two components, and their respective vertex sets are the two partite sets of G.

Figure 2 shows three graphs H, K and $H \times K$ (solid) together with their Boolean squares H^s, K^s and $(H \times K)^s$ (dashed). Notice that $(H \times K)^s = H^s \times K^s$. This is in fact a general principle: Lemma 5.33 of [7] states that $(G_1 \times G_2 \times \cdots \times G_k)^s = G_1^s \times G_2^s \times \cdots \times G_k^s$.

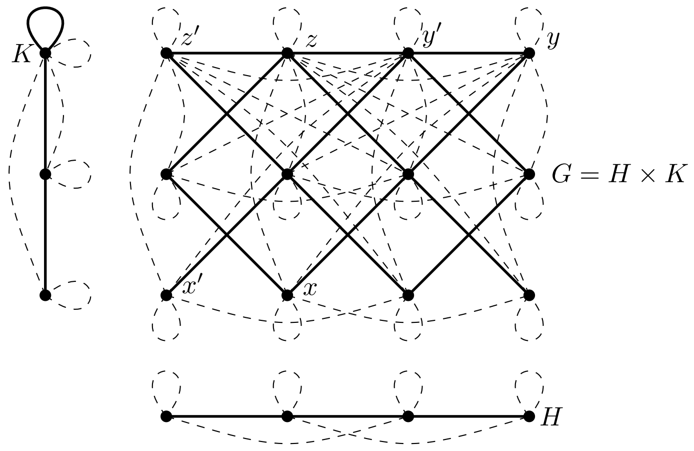


Figure 2: Graphs H, K and $G = H \times K$ (solid) and their Boolean squares (dashed)

The Cartesian Skeleton

We now show how to remove strategic edges from G^s to obtain the Cartesian skeleton $S(G)$.

Given a factorization $G = H \times K$, we say that an edge $(h, k)(h', k')$ of G^s is **Cartesian** relative to the factorization $H \times K$ if either $h = h'$ and $k \neq k'$, or $h \neq h'$ and $k = k'$. For example, in Figure 2 edges xz and zy of G^s are Cartesian (relative to the factorization $H \times K$), but edges xy and yy of G^s are not Cartesian. We now identify three intrinsic criteria for edges of G^s that tell us if they may fail to be Cartesian relative to some factoring of G.

1. If xy is a loop (i.e. if $x = y$) then xy cannot be Cartesian.
2. In Figure 2 edge xy of G^s is not Cartesian. For this edge of G^s there is a $z \in V(G)$ with $N_G(x) \cap N_G(y) \subset N_G(x) \cap N_G(z)$ and $N_G(x) \cap N_G(y) \subset N_G(y) \cap N_G(z)$.
3. In Figure 2 the edge $x'y'$ of G^s is not Cartesian. For this edge of G^s there is a $z \in V(G)$ with with $N_G(x') \subset N_G(z') \subset N_G(y')$.

Our aim is to remove from G^s all edges that meet one of these criteria. Observe that the second and third criteria are somewhat interdependent. For example, $N_G(x) \subset N_G(z) \subset N_G(y)$ implies $N_G(x) \cap N_G(y) \subset N_G(y) \cap N_G(z)$. Also, $N_G(y) \subset N_G(z) \subset N_G(x)$ implies $N_G(x) \cap N_G(y) \subset N_G(x) \cap N_G(z)$. This allows us to package the above three conditions into the following definition.

Definition 2.1. An edge xy of G^s is **dispensable** if $x = y$ or there exists $z \in V(G)$ for which both of the following statements hold.

1. $N_G(x) \cap N_G(y) \subset N_G(x) \cap N_G(z)$ or $N_G(x) \subset N_G(z) \subset N_G(y)$
2. $N_G(y) \cap N_G(x) \subset N_G(y) \cap N_G(z)$ or $N_G(y) \subset N_G(z) \subset N_G(x)$.

Several comments are in order. First, in this paper the symbol $\subset$ means *proper* containment, that is $A \subset B$ means $A \subseteq B$ and $A \neq B$. Also note that the above statements (1) and (2) are symmetric in x and y. We often rely on the following remark.

Remark 2.2. We reiterate and emphasize the following.

- Condition $N(x) \subset N(z) \subset N(y)$ in (1) implies $N(y) \cap N(x) \subset N(y) \cap N(z)$ in (2).
- Condition $N(y) \subset N(z) \subset N(x)$ in (2) implies $N(x) \cap N(y) \subset N(x) \cap N(z)$ in (1).
- Thus non-loop edge xy is dispensable if and only if $N(x) \subset N(z) \subset N(y)$ or $N(y) \subset N(z) \subset N(x)$, or both $N(x) \cap N(y) \subset N(x) \cap N(z)$ and $N(y) \cap N(x) \subset N(y) \cap N(z)$.
- Furthermore, if xy is not dispensable, then statements $N(x) \cap N(y) \subset N(x) \cap N(z)$ and $N(y) \cap N(x) \subset N(y) \cap N(z)$ cannot both be true.

Definition 2.3. The **Cartesian skeleton** of a graph G is the graph $S(G)$ obtained from G^s by removing all dispensable edges. To be specific, if G is a graph, let $D(G^s)$ be the set of dispensable edges of G^s. The Cartesian skeleton of G is the graph $S(G)$ with $V(S(G)) = V(G^s)$ and $E(S(G)) = E(G^s) - D(G^s)$.

Figure 3 is the same as Figure 1, except all dispensable edges of H^s, K^s and $(H \times K)^s$ are deleted. Thus the remaining dashed edges are $S(H), S(K)$ and $S(H \times K)$. Note that although $S(G)$ was defined without regard to the factorization $G = H \times K$, we nonetheless have $S(H \times K) = S(H) \Box S(K)$. The following lemma and proposition show this always holds.

The proofs make frequent use of the equation

$$N_G(h, k) \cap N_G(h', k') \quad = \quad (N_H(h) \cap N_H(h')) \times (N_K(k) \cap N_K(k')),$$

which follows from $N_G(h, k) = N_H(h) \times N_K(k)$ and simple set theory.

Lemma 2.4. *If G is an R-thin graph with an arbitrary factorization $G = H \times K$, then every edge of $S(G)$ is Cartesian with respect to this factorization.*

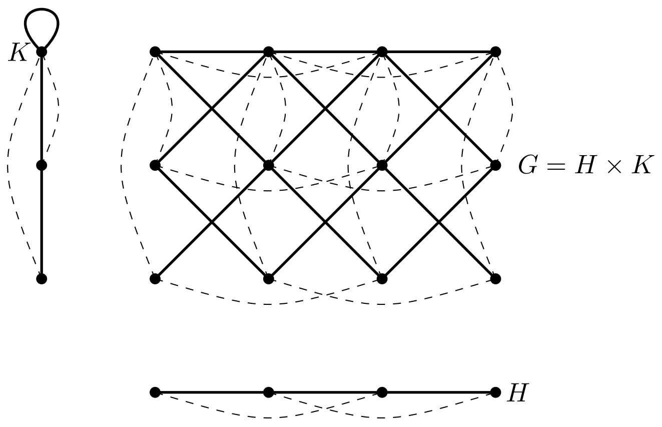


Figure 3: Graphs H, K and $H \times K$ (solid) and their Cartesian skeletons (dashed)

Proof. We show that any non-Cartesian edge of G^s is dispensible. Suppose that the edge $(h,k)(h',k')$ of G^s is not Cartesian. If it is a loop, then it is automatically dispensable. We therefore assume $h \neq h'$ and $k \neq k'$. Observe:

$$\begin{aligned} N_G(h,k) \cap N_G(h',k') &= \big(N_H(h) \cap N_H(h')\big) \times \big(N_K(k) \cap N_K(k')\big) \\ &\subseteq N_H(h) \times \big(N_H(k) \cap N_H(k')\big) \\ &= N_G(h,k) \cap N_G(h,k') \end{aligned}$$

$$\begin{aligned} N_G(h',k') \cap N_G(h,k) &= \big(N_H(h') \cap N_H(h)\big) \times \big(N_K(k') \cap N_K(k)\big) \\ &\subseteq \big(N_H(h') \cap N_H(h)\big) \times N_K(k') \\ &= N_G(h',k') \cap N_G(h,k') \end{aligned}$$

If both the above inclusions are proper, then $(h,k)(h',k')$ is dispensable, and we are done. Otherwise at least one inclusion is actually equality, and we get $N_H(h) \cap N_H(h') = N_H(h)$ in the first case or $N_K(k) \cap N_K(k') = N_K(k')$ in the second. From this, $N_H(h) \subseteq N_H(h')$ or $N_K(k') \subseteq N_K(k)$, and by R-thinness

$$N_H(h) \subset N_H(h') \quad \text{or} \quad N_K(k') \subset N_K(k). \tag{2.1}$$

Repeating the above argument but interchanging h with h', and k with k', we get that either $(h,k)(h',k')$ is dispensable (in which case we are done), or

$$N_H(h') \subset N_H(h) \quad \text{or} \quad N_K(k) \subset N_K(k'). \tag{2.2}$$

From inclusions (2.1) and (2.2) we see $N_K(h) \subset N_K(h')$ and $N_H(k) \subset N_H(k')$, or $N_K(k') \subset N_K(k)$ and $N_H(h') \subset N_H(h)$. The first case gives

$$N_H(h) \times N_K(k) \subset N_H(h) \times N_K(k') \subset N_H(h') \times N_K(k')$$

which is $N_G(h,k) \subset N_G(h,k') \subset N_G(h',k')$, so $(h,k)(h',k')$ is dispensable. The second case yields $N_G(h',k') \subset N_G(h,k') \subset N_G(h,k)$, and again $(h,k)(h',k')$ is dispensable. □

Proposition 2.5. *If H, K are R-thin graphs without isolated vertices, then $S(H \times K) = S(H) \Box S(K)$.*

Proof. Since the lemma states that all non-Cartesian edges of G^s are dispensable we can restrict attention to the Cartesian edges of G^s. We must show that an edge $(h,k)(h',k)$ of G^s is dispensable if and only if hh' is dispensable in H^s. (By symmetry this implies that $(h,k)(h,k')$ of G^s is dispensable if and only if kk' is dispensable.)

Suppose hh' is dispensable in H^s. Then there exists an z' in $V(H)$ such that both of the following conditions are satisfied.

$$\begin{array}{lcl} N_H(h) \cap N_H(h') \subset N_H(h) \cap N_H(z') & \text{or} & N_H(h) \subset N_H(z') \subset N_H(h') \\ N_H(h') \cap N_H(h) \subset N_H(h') \cap N_H(z') & \text{or} & N_H(h') \subset N_H(z') \subset N_H(h) \end{array} \tag{2.3}$$

Since $N_K(k) \neq \emptyset$ (there are no isolated vertices) we can multiply each set in (2.3) by $N_K(k)$ on the right side and still preserve the proper inclusions. Then the observation $N_H(u) \times N_K(v) = N_{H \times K}(u,v)$ shows that the dispensability conditions (1) and (2) are satisfied for $x = (h,k), y = (h',k)$ and $z = (z',k)$.

Conversely, suppose $(h,k)(h',k)$ is dispensable in $(H \times K)^s$, so there is a vertex $z = (z',z'')$ in $H \times K$ such that the dispensability conditions (1) and (2) are satisfied for $x = (h,k), y = (h',k)$ and $z = (z',z'')$. The various cases are considered below.

If $N_G(x) \subset N_G(z) \subset N_G(y)$, i.e. if $N_H(h) \times N_K(k) \subset N_H(z') \times N_K(z'') \subset N_H(h') \times N_K(k)$, then $N_K(z'') = N_K(k)$. Then the fact that $N_K(k) \neq \emptyset$ permits cancellation of the common factor $N_K(k)$, so $N_H(h) \subset N_H(z') \subset N_H(h')$, and hh' is dispensable. We reach the same conclusion if $N_G(y) \subset N_G(z) \subset N_G(x)$.

Finally, suppose there is a $z = (z',z'')$ with both $N_G(x) \cap N_G(y) \subset N_G(x) \cap N_G(z)$ and $N_G(y) \cap N_G(x) \subset N_G(y) \cap N_G(z)$. Rewrite this as

$$\begin{array}{lcl} N_G(h,k) \cap N_G(h',k) & \subset & (N_G(h,k) \cap N_G(z',z'') \\ N_G(h',k) \cap N_G(h,k) & \subset & (N_G(h',k) \cap N_G(z',z''), \end{array}$$

which is the same as

$$\begin{array}{lcl} (N_H(h) \cap N_H(h')) \times N_K(k) & \subset & (N_H(h) \cap N_H(z')) \times (N_K(k) \cap N_K(z'')) \\ (N_H(h') \cap N_H(h)) \times N_K(k) & \subset & (N_H(h') \cap N_H(z')) \times (N_K(k) \cap N_K(z'')). \end{array}$$

Given that $N_K(k) \not\subset N_K(k) \cap N_K(z'')$, the following must hold.

$$\begin{array}{lcl} N_H(h) \cap N_H(h') & \subset & N_H(h) \cap N_H(z') \\ N_H(h') \cap N_H(h) & \subset & N_H(h') \cap N_H(z') \end{array}$$

Thus hh' is dispensable. □

We next consider connectivity of $S(G)$. The following lemma is needed.

Lemma 2.6. *If $x, y \in V(G)$ and $N(x) \subset N(y)$, then G^s has an x-y path consisting of non-dispensable edges.*

Proof. We use induction on the largest integer k for which there is a chain

$$N(x) \subset N(z_1) \subset N(z_2) \subset N(z_3) \subset \ldots \subset N(z_k) \subset N(y). \tag{2.4}$$

Note that k is the number of intermediate subsets between $N(x)$ and $N(y)$, and can thus be zero. We claim that if $k = 0$, then xy is a non-dispensable edge of G^s. Certainly $N(x) \subset N(y)$ implies $xy \in E(G^s)$. Also there is no z with $N(x) \cap N(y) \subset N(x) \cap N(z)$, for if there were, the condition $N(x) \subset N(y)$ would yield $N(x) \subset N(x) \cap N(z)$, which is impossible. Finally, as $k = 0$, there is no z for which $N(x) \subset N(z) \subset N(y)$. It follows that xy is a non-dispensable edge of G^s.

Suppose $k > 0$ and that G^s has a u-v path of non-dispensable edges whenever $N(u) \subset N(v)$ and the longest chain $N(u) \subset N(w_1) \subset N(w_2) \ldots \subset N(v)$ has fewer than k intermediate sets $N(w_i)$. Then the chain (2.4) can be broken into two maximal chains $N(x) \subset N(z_1)$ and $N(z_1) \subset N(z_2) \subset N(z_3) \subset \ldots \subset N(y)$, each with fewer than k intermediate subsets. By the inductive hypothesis there are x-z_1 and z_1-y paths of non-dispensable edges in G^s. □

Proposition 2.7. *Suppose G is connected.*

1. *If G has an odd cycle, then $S(G)$ is connected.*
2. *If G is bipartite, then $S(G)$ has two components whose respective vertex sets are the two partite sets of G.*

Proof. Observe that the statement is true if $S(G)$ is replaced by G^s. As $S(G)$ is just G^s with the dispensable edges removed, we need only prove that for any dispensable edge $xy \in E(G^s)$, there is an x-y path in G^s consisting of non-dispensable edges.

For each edge $xy \in E(G^s)$, define the integer

$$k_{xy} = \max\{\, |N(u) \cap N(v)| - |N(x) \cap N(y)| \;:\; u, v \in V(G), u \neq v\}.$$

Notice $k_{xy} \geq 0$. (Put $u = x$ and $v = y$.) If $k_{xy} = 0$, then the definition of k_{xy} implies that there is no z for which $N(x) \cap N(y) \subset N(x) \cap N(z)$ and $N(y) \cap N(x) \subset N(y) \cap N(z)$. Therefore xy is not dispensable if $k_{xy} = 0$.

Take $N > 0$, and assume that whenever G^s has a dispensable edge xy with $k_{xy} < N$ there is a x-y path in G^s composed of non-dispensable edges. Now suppose xy is dispensable and $k_{xy} = N$. If $N(x) \subset N(y)$ or $N(y) \subset N(x)$, then we are done, by Lemma 2.6, so assume $N(x) \not\subset N(y)$ and $N(y) \not\subset N(x)$. As xy is dispensable, there is a vertex z with

$$\begin{cases} N(x) \cap N(y) \subset N(x) \cap N(z) \\ N(y) \cap N(x) \subset N(y) \cap N(z). \end{cases}$$

This implies $N(x) \cap N(z) \neq \emptyset \neq N(y) \cap N(z)$, so $xz, yz \in E(G^s)$. But it also means

$$|N(u) \cap N(v)| - |N(x) \cap N(z)| \; < \; |N(u) \cap N(v)| - |N(x) \cap N(y)|$$

for all u, v, so $k_{xz} < k_{xy}$. Similarly $k_{zy} < k_{xy}$. The induction hypothesis gurantees there are x-z and z-y paths of non-dispensable edges in G^s. □

Since $S(G)$ is defined entirely in terms of the adjacency structure of G, we have the following immediate consequence of the definition.

Proposition 2.8. *Any isomorphism $\varphi : G \to H$, as a map $V(G) \to V(H)$, is also an isomorphism $\varphi : S(G) \to S(H)$.*

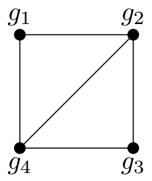
Prime Factor Decompositions

We pause to briefly mention how these results tie in to the existing literature concerning unique prime factor decompositions over the direct product. Suppose there is an isomorphism $C \times D \to A \times B$, of R-thin connected non-bipartite graphs. By the results of the previous section this induces an isomorphism $S(C)\Box S(D) \to S(A)\Box S(B)$. Using unique factorization properties for the Cartesian product we can factor $A = A'_C \Box A'_D$ and $B = B'_C \Box B'_D$ so that the isomorphism $S(C)\Box S(D) \to A'_C \Box A'_D \Box B'_C \Box B'_D$ has form $(c,d) \mapsto (\alpha_C(c), \alpha_D(d), \beta_C(c), \beta_D(d))$. Then, as in Lemma 5.41 of [7], we construct a "common refinement" $A \times B = A_C \times A_D \times B_C \times B_D$ where $A = A_C \times A_D$ and $B = B_C \times B_D$ and $C \cong A_C \times B_C$ and $D \cong A_D \times B_D$. From this it follows that any connected R-thin non-bipartite graph has a unique prime direct product factorization (Lemma 5.38 of [7]). It is then not difficult to show (Theorem 5.4 of [7]) that the same result holds when we remove the condition of R-thinness.

In carrying out this procedure algorithmically, it is of course essential that $S(G)$ can be computed efficiently. We now turn our attention to this task.

Algorithmic Construction of $S(G)$

Here we describe polynomial algorithms that construct $S(G)$. For this we use an adjacency list data structure for G. Fix an indexing $V(G) = \{g_1, g_2, \ldots, g_n\}$. Represent G as a table with n rows indexed by the vertices $g_1, g_2, \ldots, g_n$. Row i contains a list of the neighbors of g_i. This is illustrated in Figure 4.



vertex	neighbors
g_1	g_2, g_4
g_2	g_1, g_3, g_4
g_3	g_2, g_4
g_4	g_1, g_2, g_3

Figure 4: Adjacency list representation of G

If we wanted to check whether $g_j \in N_G(g_i)$, in constant time, then we could use an adjacency matrix. However, this would require n^2 space. We now show how to use a trick from [2, Exercise 12.1-4] to check that $g_j \in N_G(g_i)$ in constant time, after some preprocessing of $O(\deg(g_i))$ time and $O(n)$ space.

First form a reference vector c_i of length n, where every $c_i(k)$ is an uninitialized pointer. Next form a vector ℓ_i of length $\deg(g_i)$, where every entry is a pointer. For each g_k in the adjacency list for g_i, do the following. If g_k is the pth vertex on the adjacency list for g_i, set $c_i(k)$ to point to $\ell_i(p)$, and set $\ell_i(p)$ to point back to $c_i(k)$. Then $g_j \in N(g_i)$ if and only if $c_i(j)$ points to a pointer in ℓ_i that points back to $c_i(j)$. This can be checked in constant time, while the effort to create c_i and ℓ_i takes $O(\deg(g_i))$ time and $O(n)$ space.

For $x = g_i$ we will also write c_x and ℓ_x instead of c_i and ℓ_i.

The next two remarks are central ideas in our algorithms.

Remark 2.9. Once c_i and ℓ_i have been created and linked, we can form a list representation of any intersection $N(g_i) \cap N(g_j)$ in $O(\deg(g_j))$ time as follows. Begin with an empty list I. Then for each $x \in N(g_j)$ check whether $x \in N(g_i)$, and if so, then append it to I.

Remark 2.10. If X and Y are finite sets, then $X \subset Y$ means $|X| = |X \cap Y|$ and $|X| < |Y|$. For instance, $N(x) \subset N(z)$ provided $|N(x)| = |N(x) \cap N(z)|$ and $|N(x)| < |N(z)|$. Similarly $N(x) \cap N(y) \subset N(x) \cap N(z)$ if $|N(x) \cap N(y)| = |N(x) \cap N(y) \cap N(z)|$ and $|N(x) \cap N(y)| < |N(x) \cap N(z)|$.

Thus we can decide whether an edge xy meets conditions (1) and (2) for dispensability (in Definition 2.1) by making such comparisons among numbers $|N(x)|$, $|N(y)|$, $|N(z)|$, $|N(x) \cap N(y)|$, $|N(x) \cap N(z)|$, $|N(y) \cap N(z)|$, and $|N(x) \cap N(y) \cap N(z)|$.

Proposition 2.11. *Given an edge xy of G^s together with the reference vectors c_x, ℓ_x and c_y, ℓ_y, we can check the validity of dispensability conditions (1) and (2) for any vertex $z \in V(G) - \{x, y\}$ in $O(\deg(z))$ time.*

Proof. We can assume that $z \neq y$ and compute the following sets and numbers, using c_x, ℓ_x and c_y, ℓ_y whenever appropriate:

(i) $|N_G(z)|$

(ii) $N_G(x) \cap N_G(z)$ and $|N_G(x) \cap N_G(z)|$

(iii) $N_G(y) \cap N_G(z)$ and $|N_G(y) \cap N_G(z)|$

(iv) $|N_G(x) \cap N_G(y) \cap N_G(z)|$

By comparing the cardinalities of the intersections computed above, we check if conditions (1) and (2) from Definition 1 hold for the dispensability of xy. (See Remark 2.10.) All computations can be executed in $O(\deg(z))$ time. □

We will present two algorithms for the computation of $S(G)$, both based on the above remarks. The first considers all triples of distinct vertices x, y, z.

Algorithm 2.12.

Input: Adjacency list representation for graph G with n vertices.
Output: Adjacency list representation for $S(G)$.

```
1 For all pairs of distinct vertices $x$,$y$ of $G$ do:
    1.1 Compute $c_x$, $\ell_x$, $c_y$, $\ell_y$ and check whether $xy \in G^s$. If not, then continue with
        the next pair $x, y$.
    1.2 For all $z \in V(G) - \{x, y\}$ check the validity of the dispensability conditions.
    1.3 If the dispensability conditions fail for every $z$, add $xy$ to the adjacency list of
        $S(G)$.
2 Return the adjacency list representation for $S(G)$.
```

Proposition 2.13. *Given an input graph G of size m and order n, the time complexity of Algorithm 2.12 to compute $S(G)$ is $O(mn^2)$. The space complexity is determined by the size of the output, that is, the number of edges in $S(G)$. It is between $O(n)$ and $O(n^2)$.*

Proof. Step 1.1 takes $O(n)$ time. By Proposition 2.11, Step 1.2 is $O(\deg(z))$ for every z, contributing to a total of $\sum_{z \in G} O(\deg(z)) = O(m)$ time. Step 1.3 takes constant time. Since we have to perform Steps 1.1–1.3 at most n^2 times, we arrive at the asserted time complexity. □

In the above algorithm we took all pairs x, y and checked in Step 1.1 whether they were in $E(G^s)$. For the pair x, z the check occurs in Step 1.2.

The next algorithm makes use of the fact that conditions (1) and (2) for dispensability in Definition 2.1 can hold only if y and z are both at distance two from x. (We must have $d_G(x, y) = 2$ for $xy \in E(G^s)$, and $d_G(x, z) \neq 2$ implies $N_G(x) \cap N_G(z) = \emptyset$, whence none of the conditions hold. In fact, by condition (2), there must be a neighbor of x, say y', that is adjacent to both y and z.) The algorithm let y and z run through the sets of neighbors of neighbors of x. To reach all vertices y of distance 2 from x we thus consider all neighbors y' of x and then all neighbors y of the y'. Since it may be possible to reach y from x on many distinct paths of length 2, and since we do not know all paths a priori, the complexity of this method may be high. However, if $\Delta^3 < n^2$, then it is better than $O(mn^2)$, as we shall see.

Note that we already observed that we can assume the existence of a $y' \in N(x)$ that is a neighbor of both y and z.

Algorithm 2.14.

Input: Adjacency list representation for graph G with n vertices.
Output: Adjacency list representation for $S(G)$.

```
1 For each $x \in V(G)$ do:
    1.1 Compute $c_x$, $\ell_x$ and $|N_G(x)|$.
    1.2 For every $y' \in N_G(x)$ do:
        1.2.1 For all $y \in N_G(y') - \{x\}$ do:
            1.2.1.1 Compute $c_y$, $\ell_y$ and $|N_G(y)|$.
            1.2.1.2 For all $z \in N_G(y') - \{x, y\}$ do:
                1.2.1.2.1 Check the dispensability conditions (1) and (2).
            1.2.1.3 If dispensability conditions fail for every $z$, add $xy$ to the adja-
                cency list of $S(G)$.
2 Return the adjacency list representation for $S(G)$.
```

Proposition 2.15. *Given an input graph G of size m, order n and maximum degree Δ, the time complexity of Algorithm 2.14 to compute $S(G)$ is $O(m\Delta^3)$. The space complexity is between $O(n)$ and $O(n^2)$.*

Proof. We have loops for x, y', y, z, and show first that the complexity is given by the expression

$$\sum_{x \in V(G)} \left\{ O(|N(x)|) + \sum_{y' \in N(x)} \sum_{y \in N(y')} \left[O(|N(y)|) + \left(\sum_{z \in N(y')} O(|N(z)|) \right) + O(1) \right] \right\}.$$

The sum over $x \in V(G)$ stands for the Loop 1 in the algorithm, and the term $O(|N(x)|)$ expresses the contribution of Step 1.1.

The next step in the x-loop is 1.2. It is a sum over all neighbors of x. For every such neighbor y' we have to execute 1.2.1. It is a loop for all $y \in N(y')$. Every instance consists of three subinstances. The cost for 1.2.1.1 is $O(|N(y)|)$, the one for 1.2.1.3 is $O(1)$, whereas 1.2.1.2 is a sum over $z \in N(y')$, every instance contributing a cost of $O(|N(z)|)$.

To evaluate this expression, note that the term 1.2.1.2 is nested in all loops and has the largest contribution to the complexity. It therefore suffices to evaluate just the expression

$$\sum_{x\in V(G)} \sum_{y'\in N(x)} \sum_{y\in N(y')} \sum_{z\in N(y')} O(|N(z)|).$$

Clearly $\sum_{y\in N(y')} \sum_{z\in N(y')} O(|N(z)|) = O(\Delta^3)$. Thus the total value is

$$\sum_{x\in V(G)} \sum_{y'\in N(x)} O(\Delta^3) = O(\Delta^3) \sum_{x\in V(G)} \sum_{y'\in N(x)} 1 = O(\Delta^3) \sum_{x\in V(G)} |N(x)| = O(\Delta^3)2m.$$

$\square$

Note that $O(m\,\Delta^3)$ is a better bound than $O(mn^2)$ when $\Delta^3 < n^2$, and that $m\Delta^3$ can be close to n for sparse graphs, say direct products of cycles or of cubic graphs.

3 The Cartesian Skeleton for the Strong Product

We now describe a variation on $S(G)$, which we denote as $S[G]$. This modified skeleton is a subgraph of G (*not* of G^s) having the property $S[H \boxtimes K] = S[H]\Box S[K]$. We define it almost exactly as we defined $S(G)$, except that we use closed neighborhoods instead of open neighborhoods, and it is a subgraph of G rather than G^s.

We say an edge xy of a graph $G \in \Gamma$ is *dispensable* if there exists $z \in V(G)$ for which both of the following statements hold.

(1-strong) $N_G[x] \cap N_G[y] \subset N_G[x] \cap N_G[z]$ or $N_G[x] \subset N_G[z] \subset N_G[y]$

(2-strong) $N_G[y] \cap N_G[x] \subset N_G[y] \cap N_G[z]$ or $N_G[y] \subset N_G[z] \subset N_G[x]$.

The **closed Cartesian skeleton** of G is the graph $S[G]$ obtained from G by removing all dispensable edges.

From here we easily obtain analogues of Lemmas 2.4 and 2.6 and Propositions 2.5 and 2.7. In the proofs one simply replaces open neighborhoods with closed neighborhoods. We use $N_{H\boxtimes K}[(h,k)] = N[h] \times N[k]$ instead of $N_{H\times K}(h,k) = N(h) \times N(k)$. We also replace the condition $(N(x) = N(y)) \implies (x = y)$ for R-thinness with with the condition $(N[x] = N[y]) \implies (x = y)$ for S-thinness. Reasoning exactly as we did for $S(G)$ we obtain the following results for $S[G]$. (The only substantial difference is that for connectivity we no longer need G to be non-bipartite, as $S[G]$ is a subgraph of G, not of G^s. We can also remove the condition that G have no isolated vertices, since $N[x] \neq \emptyset$, even if x is isolated.)

Proposition 3.1. *If G is connected, then $S[G]$ is connected. Also, $S[H\boxtimes K] = S[H]\Box S[K]$ for S-thin graphs.*

We now adapt Algorithm 2.12 to $S[G]$. Note that the complexities of computing intersections of closed neighborhoods are the same as those for open neighborhoods. We use the same notation c_i and ℓ_i for the reference vectors for closed neighborhoods.

We remark that conditions (1-strong) and (2-strong) for dispensability can hold only if y and z are both in $N[x]$. (We must have $y \in N(x)$ in order that $xy \in E(G)$, and $z \notin N[x]$ implies $N[x] \cap N[y] \not\subset N[x] \cap N[z]$, whence none of the conditions hold.) In other words, the dispensability conditions can hold only if x, y and z induce a triangle in G. Thus in checking for dispensability of xy, the algorithm needs to consider only those z in $N[x]$.

We will also make use of the analog of Proposition 2.11 for strong products. Its validity is obvious by the above remarks.

Proposition 3.2. *Given distinct vertices x,y in $V(G)$ together with reference vectors c_x, ℓ_x and c_y, ℓ_y, we can check the validity of dispensability conditions (1-strong) and (2-strong) for any vertex $z \in V(G) - \{x, y\}$ in $O(\deg(z))$ time.*

Algorithm 3.3.

Input: Adjacency list representation for graph G with n vertices.
Output: Adjacency list representation for $S[G]$.

```
1 For each edge $xy \in E(G)$ do:
    1.1 Compute $c_x$, $\ell_x$, $c_y$, and $\ell_y$.
    1.2 For each $z \in N(x)$ check the validity of the dispensability conditions.
    1.3 If these conditions fail for all $z$, add $xy$ to the adjacency list of $S[G]$.
2 Return the adjacency list representation for $S[G]$.
```

Proposition 3.4. *If graph G has m edges, and maximum degree Δ, then the complexity of using Algorithm 3.3 to compute $S[G]$ is the minimum of $O(m^2)$ and $O(m\Delta^2)$.*

Proof. Every instance of Loop 1 has three subinstances. The first takes $O(\Delta)$ time, and the last constant time. We will bound the second in two ways.

On one hand, for every z the cost of checking dispensability is $O(|N(z)|)$. Since $z \in N(x)$ and $N(x) \subseteq V(G)$, the time for Loop 1.2 is bounded by $\sum_{z \in V(G)} O(|N(z)|) = O(m)$. Hence every instance of Loop 1 takes $O(\Delta) + O(m) + O(1)$ time. Since there are m edges we arrive at a total complexity of $O(m^2)$.

On the other hand, the z are among the at most Δ neighbors of x, so the time needed for every z is bounded by $O(|N(z)|) = O(\Delta)$. This yields the bound of $O(\Delta^2)$ for 1.2, and a total of $O(m\Delta^2)$.

□

A Bound Involving Arboricity

The *arboricity* $a(G)$ of a graph G is the minimum number of forests into which its edges can be partitioned. It is a measure of density of G. By a theorem of Nash-Williams [11, 7] the arboricity equals

$$a(G) = \max_{H \subseteq G} \left\lceil \frac{|E(H)|}{|V(H)| - 1} \right\rceil,$$

where the maximum is taken over all nontrivial graphs $H \subseteq G$.

Observe that Δ is an upper bound of $a(G)$ for graphs with at least one edge. Clearly the arboricity of a tree T is 1, whereas its maximum degree Δ can be as large as $|V(T)| - 1$. Hence $a(G)$ can be significantly smaller than Δ.

In the remainder of the paper we will present a variant of Algorithm 3.3 for the computation of $S[G]$, having complexity $O(m\,a(G)\,\Delta)$. In order to do this we make use of a result of Chiba and Nishizeki [1, 7], which states that all triangles of a graph G can be listed in $O(m\,a(G))$ time and space. We also use the fact—noted above—that dispensability conditions (1-strong) and (2-strong) can hold only if x, y, z lie on a triangle.

Algorithm 3.5.

Input: Adjacency list representation for graph G with n vertices.
Output: Adjacency list representation for $S[G]$.

```
1 Compute all triangles of $G$ with the algorithm of Chiba and Nishizeki.
2 Initialize an empty list $t(e)$ for every edge $e \in E(G)$.
3 Scan all triangles $x_1x_2x_3$.
    3.1 For every edge $e = x_ix_j$ of this triangle add the third vertex $x_k$ to $t(e)$.
4 For each edge $xy \in E(G)$ do:
    4.1 Compute $c_x$, $\ell_x$, $c_y$, and $\ell_y$.
    4.2 For each $z \in t(xy)$ check the validity of the dispensability conditions.
    4.3 If the dispensability condition fails for every $z$, add $xy$ to the adjacency list of
        $S[G]$.
5 Return the adjacency list representation for $S[G]$.
```

A complexity analysis along the lines of the previous ones yields the following proposition.

Proposition 3.6. *If graph G has m edges, arboricity $a(G)$, and maximum degree Δ, then $S[G]$ can be computed in $O(m\,a(G)\,\Delta)$ time and $O(m\,a(G))$ space.*

4 The role of the Cartesian skeleton for prime factorization

In this section we indicate that the complexity of the computation of $S(G)$, respectively $S[G]$, determines the complexity of the prime factorization of graphs with respect to the direct, respectively the strong product. No formal proofs are given here.

As has already been pointed out in Section 2, the first step in the prime factorization of a graph is the computation of $G_R = G/R$, respectively $G_S = G/S$. This can actually be done in linear time, but even more mundane direct algorithms will not be more complex than the respective algorithms for the computation of the skeletons.

Then $S(G_R)$, respectively $S[G_S]$, is decomposed into its prime factors with respect to the Cartesian product. Since the effort to find this factorization is linear in the number of edges of the graph to be factored we remain within the time limit.

Now comes a more complex stage. Since the number of Cartesian factors of $S(G_R)$, respectively $S[G_S]$, can be larger than the number of direct, respectively strong, factors, it may be necessary to combine several Cartesian factors to get the prime direct, respectively strong, factors. In [7] all combinations of the prime factors of the Cartesian skeleton are computed for this purpose. Since the number of prime factors of a graph on n vertices is at most $\log_2 n$ the number of combinations is bounded by $2^{\log_2 n} = n$. The cost of testing every combination is the number of edges times the number of factors in G_R, resp. G_S. This leads to a complexity of $O(mn \log_2 n)$ for this step, which may be larger than the complexity bounds of our algorithms involving Δ.

Luckily one can also bound the number of factors of the Cartesian skeleton by $\log_2 \Delta^2$ in the case of the direct product and by $\log_2 \Delta$ in the case of the strong one, but this needs some argument.

From these bounds one can then obtain the complexities $O(n\Delta^2 \log_2 \Delta)$, respectively $O(m\Delta \log_2 \Delta)$, for finding the prime factors of G_R, respectively G_S, from the prime factorizations of the Cartesian skeletons.

Finally, once the prime factorizations of G_R, respectively G_S, have been found simple arithmetic suffices to find the factorizations of G.

References

[1] N. Chiba and T. Nishizeki, Arboricity and subgraph listing algorithms, *SIAM J. Comput.* **14** (1985), 210–223.

[2] T. H. Cormen, C. E. Leiserson and R. L. Rivest, *Introduction to Algorithms*, McGraw Hill, New York, 1990.

[3] W. Dörfler and W. Imrich, Über das starke Produkt von endlichen Graphen, *Österreich. Akad. Wiss. Math.-Natur. Kl. S.-B. II* **178** (1970), 247–262.

[4] J. Feigenbaum, J. Hershberger and A. A. Schäffer, A polynomial time algorithm for finding the prime factors of Cartesian-product graphs, *Discrete Appl. Math.* **12** (1985), 123–138.

[5] J. Feigenbaum and A. A. Schäffer, Finding prime factors of strong direct product graphs in polynomial time, *Discrete Math.* **109** (1992), 77–102.

[6] W. Imrich, Factoring cardinal product graphs in polynomial time, *Discrete Math.* **192** (1998), 119–144.

[7] W. Imrich and S. Klavžar, *Product Graphs; Structure and Recognition*, Wiley Interscience Series in Discrete Mathematics and Optimization, New York, 2000.

[8] W. Imrich, S. Klavžar and D. Rall, *Topics in Graph Theory: Graphs and Their Cartesian Product*, A. K. Peters Ltd., Wellesley, MA, 2008.

[9] W. Imrich and I. Peterin, Recognizing Cartesian products in linear time. *Discrete Math.* **307** (2007), 472–483.

[10] R. McKenzie, Cardinal multiplication of structures with a reflexive relation, *Fund. Math.* **70** (1971), 59–101.

[11] C. St. J. A. Nash-Williams, Decomposition of finite graphs into forests, *J. London Math. Soc.* **39** (1964), 12.

[12] G. Sabidussi, Graph multiplication, *Math Z.* **72** (1960), 446–457.

[13] V. G. Vizing, The Cartesian product of graphs (Russian), *Vyčisl. Sistemy* **9** (1963), 30–43.