Sparsest cuts and concurrent flows in product graphs

Paul Bonsma

Department of Applied Mathematics, University of Twente, P.O. Box 217, 7500 AE Enschede, Netherlands

Received 15 October 2001; received in revised form 18 September 2002; accepted 19 February 2003

Abstract

A cut \([\mathcal{S}, \overline{\mathcal{S}}]\) is a sparsest cut of a graph \(G\) if its cut value \(|\mathcal{S}| |\overline{\mathcal{S}}| / |\mathcal{S}, \overline{\mathcal{S}}|\) is maximum (this is the reciprocal of the well-known edge-density of the cut). In the (undirected) uniform concurrent flow problem on \(G\), between every vertex pair of \(G\) flow paths with a total flow of 1 have to be established. The objective is to minimize the maximum amount of flow through an edge (edge congestion). The minimum congestion value of the uniform concurrent flow problem on \(G\) is an upper bound for the maximum cut value of cuts in \(G\). If both values are equal, \(G\) is called a bottleneck graph. The bottleneck properties of cartesian product graphs \(G \times H\) are studied. First, a flow in \(G \times H\) is constructed using optimal flows in \(G\) and \(H\), and proven to be optimal. Secondly, two cuts are constructed in \(G \times H\) using sparsest cuts of \(G\) and \(H\). It is shown that one of these cuts is a sparsest cut of \(G \times H\). As a consequence, we can prove that \(G \times H\) is (not) a bottleneck graph if both \(G\) and \(H\) are (not) bottleneck graphs.

© 2003 Elsevier B.V. All rights reserved.

Keywords: Product graph; Sparsest cut; Concurrent flow; Bottleneck graph

1. Introduction

In this paper, sparsest cuts and the related concurrent flow problem are studied for cartesian product graphs (for definitions see Section 2). For basic graph theoretic terms used here we refer the reader to [6]. We will assume all graphs to be connected. If \(G = (V, E)\) is a graph and \(S \subset V\) \((S \neq V)\) is a non-empty subset of the vertices of \(G\), \([\mathcal{S}, \overline{\mathcal{S}}]\) denotes the set of edges in the edge cut induced by \(S\). These are the edges...
with one endvertex in \( S \) and one endvertex in \( \bar{S} \). Throughout this paper we will use the term ‘cut’ for edge-cuts, and indicate these cuts with the corresponding vertex set.

The density \( d(S) \) of a cut \( S \) is equal to the number of edges in the cut divided by the number of possible edges in this cut (if edges can be added):

\[
d(S) = \frac{|[S, \bar{S}]|}{|S| \parallel \bar{S}|}.
\]

Clearly, \( 0 < d(S) \leq 1 \) for all cuts \( S \). Because we study the relation of cuts to concurrent flows, the reciprocal of the density of a cut is a better measure. This value \( 1/d(S) \) is called the cut load of cut \( S \). A cut in \( G \) with maximum cut load is called a sparsest cut of \( G \). Because we focus on the cut load, we consider the optimization problem of finding a sparsest cut a maximization problem. Sparsest cuts were studied previously in \([8,9]\). Additional interesting results on sparsest cuts have appeared in the literature under other names such as minimum ratio cut (which is a generalization of the sparsest cut for network problems) or minimum quotient cut, flux or minimum edge-expansion (which are defined using a related measure on cuts).

Sparsest cuts in some sense show the weakest parts of telecommunication networks (with respect to network reliability) or the most congested parts of networks using all-to-all communication and, therefore, the problem of finding sparse cuts has applications in various network design and analysis problems (see e.g. \([2]\)). The sparsest cut problem also has many applications in the area of approximation algorithms \([7]\).

Finding a sparsest cut is NP-hard \([9]\). Accordingly, it is not likely that there are straightforward methods to prove that a cut is a sparsest cut. Fortunately, the sparsest cut problem has an approximate dual problem, namely the uniform concurrent flow problem. With this statement we mean three things: Firstly, the flow problem is formulated as a congestion minimization problem and the minimum congestion value of a flow is an upper bound for the cut load of a sparsest cut. Secondly, the ratio between the minimum value of the concurrent flow problem and the cut load of a sparsest cut is bounded \([1,7]\). Thirdly, for certain graphs, the minimum value of the concurrent flow problem is equal to the cut load of a sparsest cut, and since it is also an upper bound, it can be used to prove that a cut is a sparsest cut. Graphs which have this property are called bottleneck graphs \([9]\). Examples of bottleneck graphs are: cycles \( C_n \), trees (paths \( P_n \) in particular), complete graphs \( K_n \) and \( n \)-cubes \( Q_n \) \([8]\). Examples of non-bottleneck graphs are complete bipartite graphs \( K_{n,m} \) for \( n \geq 2 \) and \( m \geq 3 \) and expanders \([7]\).

In this paper, we will study the bottleneck properties of cartesian product graphs. Product graphs are interesting because of their applications in networks \([3,5]\). The most common product graphs in this context of networks (such as torus, hypercube and mesh networks) are products of paths and cycles. It is shown below that these product graphs are bottleneck graphs. Uniform concurrent flow in product graphs and generalizations of product graphs were previously studied in \([11]\). In \([11]\), an optimal flow in a product graph was constructed using flows in the factors. A different flow construction and optimality proof is given in this paper. The main result established in this paper is that a sparsest cut of a product graph can be directly derived from...
the sparsest cuts of its factors. The presented proof of this statement is interesting by itself. The known ways to prove that a cut is a sparsest cut is either by constructing an optimal $yXRow$ with the same value (which can only be done for bottleneck graphs) or by comparing its cut load with all other cut loads of cuts in the graph, possibly using symmetries in the graph. In our proof below, for an arbitrary product graph we construct a bottleneck graph with corresponding cuts which allows us to use a flow construction to determine the maximum cut load.

2. Terminology

For the proofs in the next section, the definitions have to be generalized for edge-weighted graphs (networks). Let $c: E(G) \rightarrow \mathbb{R}^+ \setminus \{0\}$ be a capacity function on the edges of $G$. The sum of the capacities of edges in the cut $S$ is denoted by $c[S, \bar{S}]$.

A flow $(P, f)$ on $G$ is a set of paths $P$ on $G$ and a function $f : P \rightarrow \mathbb{R}^+$. If $P$ is a set of paths, $P_{uv}$ is the subset of $P$ consisting of all paths having vertices $u,v \in V(G)$ as endvertices. (Because we consider undirected $yXRows$, $P_{uv} = P_{vu}$.) $P_e$ is the subset of $P$ consisting of all paths that contain edge $e \in E(G)$. A flow $(P, f)$ is called a uniform flow if $\sum_{p \in P_{uv}} f(p) = 1$ for all vertices $u \neq v$. The edge-load (also called edge-congestion) $\lambda(e)$ of an edge $e \in E(G)$ is defined as

$$\lambda(e) = \frac{\sum_{p \in P_e} f(p)}{c(e)}.$$ 

The network load $\lambda(P, f)$ of a flow is equal to the maximum edge-load: $\lambda(P, f) = \max_{e \in E(G)} \lambda(e)$. The goal of the uniform concurrent flow problem is to find a uniform flow that minimizes the network load. Let $\lambda(G)$ be the minimum network load over all uniform flows in $G$. A uniform flow $(P, f)$ with $\lambda(P, f) = \lambda(G)$ is called an optimal flow.

On graphs with non-uniform capacities, the cut load $\lambda(S)$ of a cut $S$ is defined as

$$\lambda(S) = \frac{|S| |\bar{S}|}{c[S, \bar{S}]}.$$ 

Note that the cut load is not defined with respect to a certain flow. It can be verified that for every uniform flow the average load of edges in a cut $S$ is at least $\lambda(S)$. Therefore, we define the cut bound $C(G)$ of $G$ as

$$C(G) = \max_{S \subset V} \lambda(S).$$ 

Note that this is a lower bound for $\lambda(G)$. Bottleneck graphs are the graphs $G$ for which $C(G) = \lambda(G)$.

To show that a certain flow is optimal, we cannot always use the cut bound. But the uniform concurrent flow problem can be formulated as a linear program (LP), so we can use its dual problem for this purpose. (For more information on linear programming and duality we recommend [4].) The following LP describes the problem of finding...
an optimal flow:

\[
\begin{align*}
\min & \quad \lambda \\
\text{s.t.} & \quad \sum_{p \in P_{uw}} f(p) \geq 1 \quad \forall u, v \in V(G), \\
& \quad c(e)\lambda - \sum_{p \in P_e} f(p) \geq 0 \quad \forall e \in E(G), \\
& \quad \lambda \geq 0, \\
& \quad f(p) \geq 0 \quad \forall p \in P.
\end{align*}
\]

For the description of the dual problem we need the following notations: Let \( t : E(G) \to \mathbb{R}^+ \) be a distance function on the edges of \( G \) (\( 0 \in \mathbb{R}^+ \)). A distance function is called a normalized distance function if \( \sum_{e \in E} c(e) t(e) = 1 \). We will only consider normalized distance functions. \( d_t(u, v) \) denotes the length of a shortest path between \( u \) and \( v \) measured over this distance function. The value of a distance function \( t \) is defined as \( \sum_{u,v \in V} d_t(u,v) \). The distance bound \( D(G) \) of \( G \) is the maximum of these values:

\[
D(G) = \max_{t : E \to \mathbb{R}^+} \sum_{u,v \in V} d_t(u,v).
\]

A distance function that gives this bound is called an optimal distance function. The following LP describes the problem of finding an optimal distance function:

\[
\begin{align*}
\max & \quad \sum_{u,v} d_t(u,v) \\
\text{s.t.} & \quad \sum_{e} c(e) t(e) \leq 1, \\
& \quad d_t(u,v) - \sum_{e,p \in P_e} t(e) \leq 0 \quad \forall u, v \in V(G) \quad \forall p \in P_{uw}, \\
& \quad d_t(u,v) \geq 0 \quad \forall u, v \in V(G), \\
& \quad t(e) \geq 0 \quad \forall e \in E(G).
\end{align*}
\]

It can be checked that the second LP is the dual of the first LP. Note that to ensure that solving the LPs gives the desired value, \( P \) has to be the set of all possible paths on the graph. This set is clearly not polynomially bounded by the size of the input and, therefore, the number of variables, respectively, inequalities of the two LPs are not polynomially bounded. There are, however, other more complicated LP-formulations of the problems without this problem [10].

Lemma 1. \( C(G) \leq D(G) = \lambda(G) \).

This is a well-known result [8]. Observe that for every cut \( S \) with a certain cut load, we can construct a normalized distance function that gives the same value: assign distance \( 1/|S, \bar{S}| \) to every edge in \( [S, \bar{S}] \). This proves the inequality. The equality between \( D(G) \) and \( \lambda(G) \) follows from the theorem of strong linear programming duality.
[4]: because we assumed $G$ to be connected, feasible solutions for both problems are easily found. Then, because the LPs are each others’ dual, both problems have optimal solutions and the value of the optimal solution of the first is equal to the value of the optimal solution of the second.

The product graph $G \times H$ of two graphs $G$ and $H$ is defined as follows: $G \times H$ has vertex set $V(G) \times V(H)$, and an edge set containing all edges of the form

$((u,x),(v,x))$

if $(u,v) \in E(G)$ and $x \in V(H)$, and

$((u,x),(u,y))$

if $(x,y) \in E(H)$ and $u \in V(G)$. Edges of the first type are called horizontal edges and edges of the second type are called vertical edges. The capacity of edges in $G \times H$ is equal to $c(u,v)$ for edges of the first type and $c(x,y)$ for edges of the second type. The subgraph of $G \times H$ induced by the vertices $(u,x)$ for a certain fixed $x \in V(H)$ and all $u \in V(G)$ is called the $G$-layer corresponding to $x$. $H$-layers corresponding to vertices in $G$ are defined analogously.

3. Results

For convenience, the theorems are only formulated for products of two graphs $G$ and $H$. We will also assume $G$ and $H$ have uniform capacities and, therefore, write $|[S,\bar{S}]|$ instead of $c[S,\bar{S}]$. It can be verified that the results which will be established can be generalized to multigraphs and, therefore, also to graphs with non-uniform capacities. Throughout this section, $n = |V(G)|$ and $m = |V(H)|$.

In the first theorem we construct a uniform flow and a normalized distance function in $G \times H$ using optimal flows and distance functions in $G$ and $H$. Lemma 1 shows us that the network load of the constructed flow and value of the distance function are equal and thus optimal.

**Theorem 2.** For any two graphs $G$ and $H$,

$$D(G \times H) = \max\{mD(G),nD(H)\}$$

$$= \lambda(G \times H) = \max\{m\lambda(G),n\lambda(H)\}.$$  

**Proof.** To prove these equalities we first show that

$$D(G \times H) \geq \max\{mD(G),nD(H)\}$$

and

$$\lambda(G \times H) \leq \max\{m\lambda(G),n\lambda(H)\}.$$  

To prove the first inequality, consider the following argument: if $t$ is an optimal normalized distance function on the edges of $G$ we can define a distance function $t'$ that is not normalized on the edges of $G \times H$ as follows:

$$t'(u,x),(v,x)) = t(u,v)$$
for every horizontal edge of \( G \times H \), and
\[ t'(u,x),(u,y) = 0 \]
for every vertical edge of \( G \times H \). Because a distance of 0 is assigned to vertical edges, shortest paths in \( G \times H \) from \((u,x)\) to \((v,y)\) have the same length as shortest paths in \( G \) from \( u \) to \( v \), regardless of the choice of \( x \) and \( y \) (if \( u = v \) then the length is 0). Therefore, a shortest path in \( G \) from \( u \) to \( v \) corresponds to \( m^2 \) shortest paths of the same length in \( G \times H \). All vertex pairs in \( G \times H \) have been considered and we have shown the following:
\[
\sum_{p,q \in V(G \times H)} d_v(p,q) = m^2 \sum_{u,v \in V(G)} d_l(u,v) = m^2 D(G).
\]
The total distance assigned to edges in \( G \times H \) is \( m \) times the total distance assigned to edges in \( G \), so to normalize the distance function we can divide all edge distances by \( m \). Then the value of the constructed distance function becomes \( mD(G) \).
A similar construction can be done using an optimal distance function on \( H \), and \( D(G \times H) \geq \max\{mD(G),nD(H)\} \) follows.
To prove the second inequality, a uniform flow in \( G \times H \) is constructed from optimal flows in \( G \) and \( H \).

The path set \( P_{uv} \) in an optimal flow \((P,f)\) in \( G \) can be used for a corresponding path set in a \( G \)-layer of \( G \times H \) from \((u,x)\) to \((v,x)\). If a flow from \((u,x)\) to \((v,x)\) is desired, this set of paths is used in the flow construction. The same can be done for a flow from \((u,x)\) to \((u,y)\).

To construct a path set from \((u,x)\) to \((v,y)\) with a total flow of 1, first we use the path set from \((u,x)\) to \((v,x)\) with a total flow of \( \frac{1}{2} \), then we use the path set from \((v,x)\) to \((v,y)\) with a total flow of \( \frac{1}{2} \). These path sets can be combined (in an arbitrary manner) to form a path set from \((u,x)\) to \((v,y)\) with a total flow of \( \frac{1}{2} \). Then the same is done using \((u,y)\) as the connection point, and together these path sets give the desired path set with a total flow of 1. Note that if \( u = v \) or \( x = y \) then the flow is not actually split into two path sets.

In every \( G \)-layer corresponding to a fixed \( x \in V(H) \) of \( G \times H \), the path set from \((u,x)\) to \((v,x)\) is used:

- \( m - 1 \) times for a flow of \( \frac{1}{2} \), once for every vertex pair \((u,y)\) and \((v,x)\) with \( y \neq x \),
- \( m - 1 \) times for a flow of \( \frac{1}{2} \), once for every vertex pair \((u,x)\) and \((v,y)\) with \( y \neq x \),
- once for a flow of 1 from \((u,x)\) to \((v,x)\).

So in every \( G \)-layer every path set is used for a total flow of \( \frac{1}{2}(m - 1) + \frac{1}{2}(m - 1) + 1 = m \). Using this flow construction the maximum load of the horizontal edges is equal to \( m\lambda(G) \). Similarly, the maximum load of the vertical edges is equal to \( n\lambda(H) \), so \( \lambda(G \times H) \leq \max\{m\lambda(G),n\lambda(H)\} \).
Now we have, using Lemma 1 for \( G \), \( H \) and \( G \times H \):
\[
D(G \times H) \geq \max(mD(G),nD(H)) \]
\[
= \max(m\lambda(G),n\lambda(H)) \geq \lambda(G \times H) = D(G \times H).
\]
Therefore, all inequalities must be equalities and the optimality of the constructed distance function and flow follows. \( \square \)

The following lemma is a similar statement for the cut bound \( C(G \times H) \), and will be proved by a construction using sparsest cuts of \( G \) and \( H \).

**Lemma 3.** For any two graphs \( G \) and \( H \),

\[
C(G \times H) \geq \max\{mC(G), nC(H)\}.
\]

**Proof.** It is shown that each cut \( S \) in \( G \) with cut load \( \lambda(S) \) corresponds to a cut \( S' \) in \( G \times H \) with cut load \( \lambda(S') = m\lambda(S) \). \( S' \) is defined as follows: \((u,x) \in S'\) if and only if \( u \in S \). Therefore \(|S'| = m|S|\) and \(|\bar{S}'| = m|\bar{S}|\). If \((u,v) \in [S, \bar{S}]\) then \(((u,x),(v,x)) \in [S', \bar{S}']\) for every \( x \in V(H) \), so \(|[S', \bar{S}']| = m|[S, \bar{S}]|\). It follows that \( \lambda(S') = (m^2/m)\lambda(S) = m\lambda(S) \). Thus \( C(G \times H) \geq mC(G) \). \( C(G \times H) \geq nC(H) \) can be shown analogously. \( \square \)

Combining Theorem 2 and Lemma 3 with Lemma 1, we have the following result.

**Corollary 4.** If \( G \) and \( H \) are bottleneck graphs, then \( G \times H \) is a bottleneck graph.

We can also conclude that if \( G \) and \( H \) are bottleneck graphs, then the cut constructed in Lemma 3 is a sparsest cut. The next theorem states that this is true in general, which allows a corollary similar to Corollary 4 to be formulated about non-bottleneck graphs.

**Theorem 5.** \( C(G \times H) = \max\{mC(G), nC(H)\} \).

**Proof.** It suffices to show that \( \lambda(S) \leq \max\{mC(G), nC(H)\} \) holds for every \( S \subset V(G \times H) \), which together with Lemma 3 proves our claim. This is done by constructing a new graph \( G' \) with non-uniform edge capacities. First, it is shown that \( C(G') = \max\{mC(G), nC(H)\} \). To conclude the proof, it is shown that every cut \( S \) in \( G \times H \) corresponds to a cut \( S' \) in \( G' \) with \( \lambda(S) \leq \lambda(S') \). For an example of the construction in this proof see Fig. 1.

For the construction of \( G' \), take two paths \( P_n \) and \( P_m \). Label the vertices of \( P_n \) (\( P_m \)) along the path with labels \( 1, \ldots, n \) (\( 1, \ldots, m \)). Set edge capacities in \( P_n \) to \( c(i,i+1) = i(n-i)/C(G) \) for \( i = 1, \ldots, n-1 \). Set edge capacities in \( P_m \) to \( c(i,i+1) = i(m-i)/C(H) \) for \( i = 1, \ldots, m-1 \). Now it can be verified that \( C(P_n) = C(G) \) and \( C(P_m) = C(H) \) (and that any edge in \( P_n \) (\( P_m \)) gives a sparsest cut). Define \( G' = P_n \times P_m \). Note that \( |V(G')| = |V(G \times H)| = nm \). Using the fact that paths are bottleneck graphs, Corollary 4 and Theorem 2 imply that \( C(G') = \max\{mC(G), nC(H)\} \).

Now we consider a cut \( S \) in \( G \times H \). We will construct a cut \( S' \) in \( G' \) with \(|S'| = |S|\) (and thus \(|\bar{S}'| = |\bar{S}|\) and \( c[S', \bar{S}'] \leq |[S, \bar{S}]|\)).

Using \( S \), we can define a cut \( X(v) \) in \( G \) for every vertex \( v \in V(H) \):

\[
X(v) = \{u \in V(G) : (u,v) \in S\}.
\]

Number the vertices \( V(H) = \{v_1, \ldots, v_m\} \) such that \( i < j \Rightarrow |X(v_i)| \geq |X(v_j)| \). Now we write \( X_i \) instead of \( X(v_i) \).
These cuts are now defined such that \( \sum_{i=1}^{m} ||[X_i, \bar{X}_i]|| \) is equal to the number of horizontal edges in the cut \([S, \bar{S}]\).

Next, we define a set of \( n \) cuts in \( H \) using \( S \):

\[
Y_k = \{v_i : k \leq |X_i|\}.
\]

Because \(|X_i|\) is decreasing, \( Y_k = \{v_1, \ldots, v_p\} \) for some \( p \).

We will prove that \( \sum_{k=1}^{n} ||[Y_k, \bar{Y}_k]|| \) does not exceed the number of vertical edges in \([S, \bar{S}]\) by constructing a mapping of the edges in these cuts to the vertical edges in \([S, \bar{S}]\) that is an injection.

Consider an edge \((v_i, v_j) \in E(H)\), suppose \( i < j \) and therefore \(|X_i| \geq |X_j|\). Between the \( G \)-layer in \( G \times H \) corresponding to \( v_i \) and the one corresponding to \( v_j \), there are at least \(|X_i| - |X_j|\) vertical edges in \([S, \bar{S}]\). Label an arbitrary subset of \(|X_i| - |X_j|\) of these edges with the numbers \(|X_j| + 1, \ldots, |X_i|\). This labeling of vertical edges is done for every edge in \( H \). The mapping is as follows: if \((v_i, v_j) \in [Y_k, \bar{Y}_k]\), then w.l.o.g. \(|X_i| \geq k \) and \(|X_j| < k\), so this edge can be mapped to the edge \(((u, v_i), (u, v_j))\) that was labeled with label \( k \). Now for every edge in \( \bigcup_{k=1, \ldots, n} [Y_k, \bar{Y}_k] \) a unique edge in \([S, \bar{S}]\) is assigned, which proves our claim.

Next, we will construct a cut \( S' \) in \( G' \):

\[
S' = \{(k, i) : k \leq |X_i|\},
\]
so the horizontal edges in \([S', S']\) (corresponding to edges of \(P_n\)) are of the form \(((|X_i|, i), (|X_i| + 1, i))\). Using the definition of \(Y_k\), we can rewrite \(S'\) as

\[
S' = \{(k, i) : v_i \in Y_k\}
\]

and using the fact that \(Y_k = \{v_1, \ldots, v_p\}\) for some \(p\), we know that the vertical edges in \([S', S']\) (corresponding to edges of \(P_m\)) are of the form \(((k, |Y_k|), (k, |Y_k| + 1))\). Now we have

\[
[S', S'] = \{((|X_i|, i), (|X_i| + 1, i)) : |X_i| \neq 0 \land |X_i| \neq n\}
\]

\[
\cup \{((k, |Y_k|), (k, |Y_k| + 1)) : |Y_k| \neq 0 \land |Y_k| \neq m\}.
\]

The capacity of the cut \([S', S']\) is equal to

\[
c[S', S'] = \sum_{i=1}^{m} \frac{|X_i|(n - |X_i|)}{C(G)} + \sum_{k=1}^{n} \frac{|Y_k|(m - |Y_k|)}{C(H)}
\]

\[
\leq \sum_{i=1}^{m} [|X_i, \bar{X}_i|] + \sum_{k=1}^{n} [|Y_k, \bar{Y}_k|] \leq [|S, \bar{S}|].
\]

The first equality follows from the definition of the capacities in \(G'\), the first inequality follows from the fact that \(C(G) \geq |X_i|(n - |X_i|)/[|X_i, \bar{X}_i|]\) for any \(i\) and a similar statement for \(C(H)\), and the last inequality follows from the proofs above.

Now we have proved that for any cut \(S\) in \(G \times H\), there is a cut \(S'\) in \(G'\) with

\[
\lambda(S) = \frac{|S||S|}{[S, \bar{S}]} \leq \frac{|S'|||S'|}{c[S', S']} \leq \max(mC(G), nC(H)),
\]

so \(C(G) \leq \max(mC(G), nC(H))\).

This proves that the cut constructed in Lemma 3 is a sparsest cut of \(G \times H\). So every graph \(G \times H\) has a sparsest cut that consists only of horizontal or only of vertical edges. We also have the following corollary.

**Corollary 6.** If \(G\) and \(H\) are not bottleneck graphs, then \(G \times H\) is not a bottleneck graph.

In view of Corollaries 4 and 6, there is one question left: what if one graph (say \(G\)) is a bottleneck graph and the other graph (\(H\)) is not? Interestingly, the result only depends on the values of \(D(G)\) and \(D(H)\), not on \(C(G)\) and \(C(H)\):  

**Corollary 7.** If \(G\) is a bottleneck graph and \(H\) is not a bottleneck graph, then \(G \times H\) is a bottleneck graph if and only if \(nD(H) \leq mD(G)\).
Proof. If $nD(H) > mD(G)$, then
\[
D(G \times H) = \max\{mD(G), nD(H)\} = nD(H) > \max\{mC(G), nC(H)\} = C(G \times H),
\]
so $G \times H$ is not a bottleneck graph.

If $nD(H) \leq mD(G)$, then we know that $nC(H) < nD(H) \leq mD(G) = mC(G)$, so
\[
C(G \times H) = \max\{mC(G), nC(H)\} = mC(G)
\]
\[
= mD(G) = \max\{mD(G), nD(H)\} = D(G \times H)
\]
and $G \times H$ is a bottleneck graph. □

References