Parallel Construction of Perfect Matchings and Hamiltonian Cycles on Dense Graphs

ELIAS DAHLHAUS

AND

MAREK KARPINSKI*

DEPT. OF COMPUTER SCIENCE
UNIVERSITY OF BONN

Abstract.

The intensive study of fast parallel and distributed algorithms for various routing (and communications) problems on graphs with good expanding properties [PU1], [PU2], [SS] is being carried out recently. The parallel solutions for expanders that already exist [PU 2] required an extensive randomization and an application of a randomized subroutine for the Maximum Matching Problem. In this paper we attack the problem of fast parallel algorithms for constructing Perfect Matchings and Hamiltonian Cycles on Dense Graph-Networks (undirected graphs of minimal degree $\frac{|V|}{2}$). Somewhat surprisingly, we design fast deterministic parallel algorithms for constructing both the Perfect Matchings and the Hamiltonian Cycles on the Dense Graphs.

The algorithm for constructing Perfect Matchings on dense graphs works in $O(\log^2 n)$ parallel time and $O(n^8)$ processors or $O(\log^4 n)$ parallel time and $O(n^4)$ processors on a CREW-PRAM. The algorithm for constructing Hamiltonian cycles on dense graphs works in $O(\log^5 n)$ parallel time and $O(n^4)$ processors on a CREW-PRAM.

Our method of the parallel solution involves a development of new dense graph combinatorics suitable for fast parallelization.

1. Introduction.

We call graphs $G = (V, E)$ with minimal degree $\frac{|V|}{2}$ *dense*. Dirac [Di] showed that each dense graph has a Hamiltonian cycle. The proof induces a polynomial time algorithm for constructing a Hamiltonian cycle for any dense graph. From Dirac's result follows easily the theorem of König that each dense graph with an even number of vertices has a perfect

*Supported in part by Leibniz Center for Research in Computer Science and the DFG Grant KA 673/2-1

matching (see [Bo]). Our problem is to construct Hamiltonian cycles and perfect matchings in dense graphs in parallel. Although the construction of a Hamiltonian cycle in NC induces a perfect matching, we shall present separate algorithms for the perfect matching problem and the Hamiltonian cycle problem. This has two reasons: First, the algorithm for the construction of a perfect matching is simpler and it is in NC^2, while the Hamiltonian cycle algorithm is in NC^3. Second, dense matching procedures are used as subroutines for the construction of a Hamiltonian cycle. Section 2 introduces some foundations on graph theory and the parallel complexity theory. Section 3 presents a parallel algorithm for the perfect matching problem, and section 4 presents a parallel algorithm for the general construction of a Hamiltonian cycle in dense graphs.

2. Basic Definitions and Notations.

A graph $G = (V, E)$ consists of a set V of vertices and a set $E \subset \{[x, y] \mid x, y \in V\}$ of (undirected) *edges*. Loops and multiple edges are not allowed. The degree $d(v)$ of a vertex v is the size of $\{x \mid [v, x] \in E\}$.

A *matching* of (V, E) is a subset $M \subset E$, s.t. any two $e_1, e_2 \in M$ have no common vertex. A *perfect* matching is a matching which covers all vertices of V. A *Hamiltonian* cycle is a cycle which passes all vertices.

NC^k is the class of functions f computable by a (logspace) uniform sequence $(C_n)_{n=1}^{\infty}$ of switching circuits, s.t.

1) for each input x of length n, $C_n(x) = f(x)$
2) The size of C_n is bounded by a polynomial
3) The depth of C_n is bounded by $O((\log n)^k$

$NC = \bigcup_{k=1}^{\infty} NC^k$

(For more details see [Co]). NC is also the class of problems computable by a PRAM (parallel random access machine) in polylog time with a polynomial processor bound. Here we use CREW-PRAMs (concurrent read/exclusive write). For more details see [Go]. For any rational number u, the largest integer $\leq u$ is denoted by $\lceil u \rceil$ and the smallest integer $\geq u$ is denoted by $\lfloor u \rfloor$.

3. Construction of a Perfect Matching for Dense Graphs.

In this section we will present an NC^2-algorithm constructing for each dense graph a perfect matching. The key of the algorithm is the following result.

Lemma 1. There is an NC^2-algorithm constructing for each graph a nonextensible (also called maximal) independent set.

Remark [Lu]: The above algorithm implemented on an EREW-PRAM (parallel random access machine without concurrent read and concurrent write) needs $O(|V|^2|E|)$ processors in the worst case. Goldberg and Spencer [GS] could diminish the number of processors on the expense of parallel time.

Lemma 2. [GS]: There is an EREW-PRAM algorithm computing a maximal independent set with $O(|E|)$ processors and a time $O((\log n)^4)$.

An immediate consequence is the following

Lemma 3.([Lu]; see also [KW]): For each graph a nonextensible matching can be constructed in NC^2.

Remark: For the construction of a nonextensible matching we need $O(|E|^4)$ processors in the worst case.

Lemma 3 can be derived from Lemma 1 by construction from a graph $G = (V, E)$ a graph $G' = (V', E')$, s.t. $V' = E$ and two edges of E are joint by an edge G' iff they have a common vertex. Clearly a nonextensible matching in G is the same as a nonextensible independent set in G'. The number of processors which are needed for the construction of a nonextensible matching can easily be derived from the construction of G'. Using the algorithm of Goldberg and Spencer, we get the following variation of Lemma 3.

Lemma 4. There is an EREW-PRAM-algorithm computing a maximal matching with $O(|E|^2)$ processors in time $O((\log n)^4)$. Now we can state the main result of this section.

Theorem 1. For each dense graph of an even number of vertices a perfect matching can be constructed in NC^2.

For the PROOF we state a straight line algorithm, s.t. each single step can be executed in NC^2.

Input: a dense graph $G = (V, E)$.

First step: Compute any nonextensible matching M_1 of G (using a Maximal Independent Set Algorithm).

Comments: *Each edge contains at least one vertex appearing in* M_1, otherwise M_1 would be extensible.
At least $\frac{|G|}{2}$ *vertices belong to an edge of* M_1.
End of Comments.

Second step: Let $\{x_1, \cdots, x_{2k}\}$ be the set of vertices of G not belonging to an edge of M_1 and define $G' = (V', E')$ as follows:
The vertex set V' consists of the edges of M_1 and of the unordered pairs $\{x_{2i-1}, x_{2i}\}$, $i = 1, \cdots, k$. The edge set is defined as follows: $\{x_{2i-1}, x_{2i}\}$ and $\{y, z\} \in M_1$ are joined by an edge in E' iff $[x_{2i-1}, y]$ and $[x_{2i}, z] \in E$ or $[x_{2i-1}, z]$ and $[x_{2i}, y] \in E$.

Comment: Note that G' is bipartite.
End of Comment.

Third Step: Compute a nonextensible matching M_2 of G'.

Comments: *Each vertex of* G' *of the form* $\{x_{2i-1}, x_{2i}\}$ *belongs to an edge of* M_2.
We shall prove this claim by the following statement.

Lemma 5. Let k be defined as above as the number of pairs $\{x_{2i-1}, x_{2i}\}$. The degree of each vertex $\{x_{2i-1}, x_{2i}\}$ is at least k.

PROOF. Assume that $\{x_{2i-1}, x_{2i}\}$ has degree ℓ. Then at most ℓ of the $|V|/2 - k$ pairs in the matching M can be touched more than twice by the edges emanating from x_{2i-1} or x_{2i}. The number of edges touching x_{2i-1} or x_{2i} and an edge $e \in M$, which is touched more than twice, can be bounded by 4ℓ.

Thus
$|V| = 2(|V|/2) \leq d(x_{2i-1}) + d(x_{2i}) \leq 4\ell + 2(|V|/2 - k - \ell) = |V| - 2(k - \ell) \leq |V| - 2$

This is a contradiction.

End of Comments.

Fourth step: for each $\{x_{2i-1}, x_{2i}\}$ as above consider the $[u, v] \in M_1$ which is joined by an edge of M_2. W.l.o.g. $[x_{2i-1}, u], [x_{2i}, v] \in E$. Delete $[u, v]$ from M_1 and add $[x_{2i-1}, u]$ and $[x_{2i}, v]$ to M_1.

Comment: M_1 is changed to a perfect matching.
End of Comment.

Last step: Output M_1.

The correctness of the algorithm follows from the comments. The algorithm defines an NC^2-function because each step is in NC^2. □

Remark: If we want to check the number of processors we have to check the number of processors of each step and take the maximum. We need a large number of processors in the first and the third step. But G' has only $|V|/2$ vertices and at most $|E|/2$ edges. Therefore the number of processors on a CREW-PRAM can be bounded by $O(|E|^4)$ for the whole algorithm if we use Luby's algorithm as a subroutine. If we use the algorithm of Goldberg and Spencer, we need a time $O(\log^4 n)$, but only $O(|E|^2)$ many processors on a CREW-PRAM. It can also be easily checked that the algorithm constructs for each graph (V, E) with an odd vertex number n and minimal degree $\lceil \frac{n}{2} \rceil$ a maximum matching of size $\lceil \frac{n}{2} \rceil$.

The next question is the parallel complexity of matching for graphs of a minimal degree $\alpha|G|$, s.t. $\alpha < \frac{1}{2}$.

Theorem 2. For $\alpha < \frac{1}{2}$ the existence problem for a perfect matching restricted to graphs $G = (V, E)$, s.t. minimal degree is $\alpha|V|$, is NC-hard for the general matching problem. This means that an NC-algorithm for the matching problem restricted to graphs of minimal degree α would deduce an algorithm for the general perfect matching problem.

PROOF. Let $G = (V, E)$ be any graph. We construct a graph $G' = (X \dot\cup Y \dot\cup V, E')$ as follows: X forms a complete subgraph of G' and Y forms an independent set in G'. Each vertex of X and each vertex of V are joined by an edge in E'. Vertices of V are joined by an edge in G' iff they are joined by an edge in G. X and Y have the same power.

Claim: G has a perfect matching iff G' has a perfect matching: Let M' be a perfect matching of G'. Then M' defines a bijection between X and Y, because Y is independent and all edges of Y go to X. Therefore no edges between V and X are in M'. That means M' restricted to V defines a perfect matching on G.

Let M be a perfect matching on G and f be a bijection between X and Y. Then clearly a perfect matching on G' is defined.

The minimal degree of G' is the power of X and by definition $|G'| = 2|X| + |G|$. Set X so large that

$\frac{|X|}{2|X|+|G|} = \alpha$

But that means

$|X| = \frac{\alpha}{(1-2\alpha)}|G|$.

But for $\alpha < \frac{1}{2}$ we have $\frac{\alpha}{1-2\alpha} > 0$ $\square$

A similar proof technique was also used by A. Broder [Br] for showing that determining the permanent of "dense" bipartite graphs is $\#P$-hard.

4. The Parallel Complexity of the Construction of a Hamiltonian Cycle.

At first we construct a maximum matching by the previous maximum matching procedure for dense graphs. There remains at most one vertex x not covered by the maximum matching. That way we get a collection of disjoint paths covering the whole graph $G = (V, E)$.

Consider now a maximum matching M on G. This can be computed in NC^2. Let $G' = (V', E')$ where $V' = M \cup \{x\}$ and $[\{x_1, x_2\}, \{y_1, y_2\}] \in E'$ iff one x_i is adjacent to a y_j. One of the $\{x_1, x_2\}$ can be equal $\{x\}$. We have to check the degree of any vertex of G'. Each adjacency of two edges of M needs at most four edges. Let d be the degree of some $e \in V'$ in G'. Then $4d + 2 > n$ if $x \notin V'$ and $4(d-1) + 4 < n$ if $x \in V'$.

In the first case n is even. Then $4d \geq n - 2$ and $2d \geq \frac{n}{2} - 1$. For the case that n is divisible by 4, $d \geq \frac{n}{4} - \frac{1}{2}$ and therefore $d \geq \frac{n}{4}$. For the case that n is not divisible by 4 $d \geq \lceil \frac{n}{4} \rceil = \lceil \frac{n/2}{2} \rceil$. Also in the latter case the perfect matching algorithm for dense graphs presents a maximum matching on G'.

In the second case we can easily see that $n \geq \frac{n+1}{4}$. Therefore $n \geq \frac{n+1}{\lfloor 4 \rfloor} \geq \frac{\lfloor \frac{n}{2} \rfloor}{\lfloor 2 \rfloor}$. We may now assume that we have at most $\frac{n}{\lfloor 4 \rfloor}$ disjoint paths.

Our aim is now to paste paths and cycles together by dense matching techniques until we get a Hamiltonian cycle.

Consider any path $p = (x_1, \cdots, x_m)$ of a dense graph.

Lemma 6. Let n be the number of vertices ($n = |V|$), a_1 be the number of edges leaving x_1 and not touching any other vertex of $p = (x_1, \cdots, x_m)$, a_m be the number of edges leaving x_m and not touching any other vertex of p. Then one of the following statements is true:

i) $a_1 + a_m > n - m$

ii) $[x_1, x_m] \in E$

iii) there is an i, s.t. $1 < i < j - 1$, s.t. $[x_1, x_m], [x_1, x_{i+1}] \in E$. $\square$

Statements ii) and iii) give us the possibility to make p a cycle.

PROOF. Assume i) and ii) are not true. Let b_1 be the number of edges $\neq \{x_1, x_2\}$ leaving x_1 and touching some $x_3 \cdots x_{m-1}$ and let b_m be the number of edges leaving x_m and touching some $x_2 \cdots x_{m-2}$.

Therefore

$a_1 + b_1 + 1 \geq \frac{n}{2}$ and $a_m + b_m + 1 \geq \frac{n}{2}$ and $a_1 + b_1 + a_m + b_m \geq n - 2$.

Now $a_1 + a_m \leq n - m$ and that means $b_1 + b_m \geq n - 2 - (a_1 + a_m) \geq m - 2 = |\{x_2, \cdots, x_{m-1}|$.

We say $(x_1, k) \in E_1 \Leftrightarrow [x_1, x_{k+1}] \in E$ and $(x_m, k) \in E_m \Leftrightarrow [x_m, x_k] \in E$. k ranges over 2 until $m-2$. iii) is satisfied if and only if one finds a k, s.t. $(x_1, k) \in E_1$ and $(x_m, k) \in E_m$. But k ranges over $m-3$ elements and $|A| := |\{k|(x_1, k) \in E_1\}| = b_1$ and $|B| := |\{k|(x_m, k) \in E_m\}| = b_m$. This means A and B have a nonempty intersection. $\square$

Now we describe the steps we want to do repeatedly and in parallel:

1) Concatenation of two paths: From $p = (x_1, \cdots, x_m)$, $q = (y_1, \cdots, y_{m_2})$, and $\{x_{m_1}, y_1\} \in E$ form $p \frown q := (x_1, \cdots, x_{m_1}, y_1, \cdots, y_{m_2})$.
2) Insertion of a path into a path or cycle: From $p = (x_1, \cdots, x_\ell), q = (y_1, \cdots, y_m)$ and $[x_i, y_i], [x_\ell, y_m] \in E$ form $p^{\text{ins}}(x_1, x_{i+1})\, q := (x_1, \cdots, x_i; y_1; \cdots, y_m, x_{i+1}, \cdots, x_\ell)$

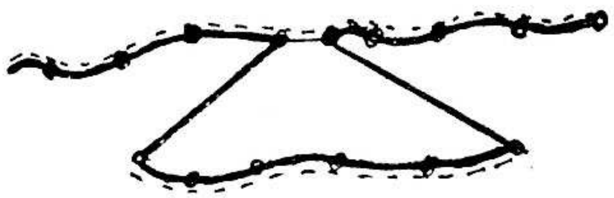

Figure 1: Insertion: The old paths are assigned by broken lines, the newly generated path is assigned by bold lines

3) Making a cycle to a path: For a cycle $C := (x_1, \cdots, x_m, x_1)$ let $C_{x_i} := (x_{i+1}, \cdots, x_m, x_1, \cdots, x_i)$.
4) Making a path to a cycle
   a) If $p := (x_n, \cdots, x_m), [x_1, x_{i+1}], [x_m, x_i] \in E$ then $p^{\text{cyc}}(x_i, x_{i+1}) := (x_1, \cdots, x_i, x_m, x_{m-1}, \cdots, x_{i+1}, x_1)$
   b) If $p = (x_n, \cdots, x_n)$ and $[x_1, x_n] \in E$, then $p^{\text{cyc}} := (x_1, \cdots, x_n, x_1)$.

Call a path p *proper* iff it suffices condition i) of the previous lemma. Otherwise we can make p a cycle by 4 a) or 4 b). We can make insertions in parallel if they use different edges $[x_i, x_{i+1}]$. This means it is sensible to make a choice for each proper path of an insertion or concatenation by maximal bipartite matching. For each path $p = (x_1, \cdots, x_n)$ let $i := i_p$ be the number of endvertices of another path, adjacent to x_1 or x_m, and $j := j_p$ be the number of path edges in which p can be inserted.

Lemma 7. Let $P := \{p_1, \cdots, p_k\}$ be a set of pairwise disjoint paths, s.t. $\bigcup_{i=1}^k p_i = V$, and (V, E) be dense, then for each proper path $p \in P$ we have $i_p + j_p \geq k$

Figure 2: Making a path a cycle. The old path is assigned by a broken line, the cycle is assigned by a bold line.

PROOF. We have $2(k-1)$ endpoints of paths not belonging to p. Let u be the number of vertices not belonging to p and a_1 be the number of vertices adjacent to x_1 and not belonging to p and a_m be the number of vertices adjacent to x_m and not belonging to p. Let ℓ be the number of path endpoints adjacent to x_1 or x_m. That means $\ell' := 2(k-1) - \ell$ is the number of path endpoints not adjacent to x_1 or x_m. We know that $a_1 + a_m > u$. But that means that the number t of vertices adjacent to x_1 and x_m not belonging to p is greater than ℓ'.

We consider vertices v adjacent to x_1 and x_m such that no path neighbour of v is adjacent to x_1 or x_m and call such a v *isolated*. We call a subpath $(u_1, \cdots, u_k)$ an *isolated chain* iff $(u_i)_{i=1}^k$ alternates between isolated vertices and vertices not adjacent to x_1 or x_m. Clearly for $u = u_1, u_k$ u is not adjacent to x_1 or x_m or u is an endpoint of any p_j. Consider maximal isolated chains and delete from them the isolated vertices which are endpoints of some p_j. Call these subpaths *alternating subpaths*. They have the property that they have at least as many vertices not adjacent to x_1 or x_2 as vertices adjacent to both x_1 and x_2. If we delete all vertices belonging to an alternating subpath, then the sum of all numbers of vertices not belonging to p adjacent to x_1 and x_m, resp., remains greater than the number i of the remaining vertices not belonging to p. Also the number of vertices adjacent to x_1 and x_m not belonging to p remains greater than ℓ'. Call the collection of these vertices A. Some $v \in A$ are at the end of some p_j. But these are at most ℓ. Call the set of the remaining vertices B and let $\ell'' = |B|$. All remaining $v \in A$ have the property that at least one path neighbour w is adjacent to x_1 or x_m. But then p can be inserted into $\{v, w\}$.

Consider subpaths $(v_1, \cdots, v_m)$, s.t. $v_2, \cdots, v_{m-1}$ are adjacent to both x_1 and x_m and v_1 and v_m are adjacent to x_1 or x_m. Each $v \in B$ is in such a subpath, and each subpath has the length of at least two vertices. Therefore we have at least $\frac{\ell''}{2}$ path edges in which p can be inserted.

But $i_p + j_p \geq \ell + \frac{\ell''}{2} > \ell + \frac{\ell'-\ell}{2} = \ell + \frac{2(k-1)-2\ell}{2} = \frac{2(k-1)}{2} = k-1$. Therefore $i_p + j_p \geq k$. $\square$

An immediate consequence of this lemma is the following: Consider the bipartite graph $\tilde{H} := (V_1 \dot{\cup} V_2, \tilde{E})$, where
$V_1 := \{p_1, \cdots, p_k\}$, $V_2 := \{x | x \text{ endpoint of some } p_k\} \cup \{[x_1, x_2] | x_1, x_2 \text{ neighbours on some } p_k\}$
and
$[p_i, x] \in \tilde{E}$ iff for one of the endpoints y of p_i, $[y, x] \in E$

and
$[p_i, [x_1, x_2]] \in \tilde{E}$ iff p_i can be inserted into $[x_1, x_2]$.

Any maximal matching M of $\tilde{H}$ (connected by an MIS-algorithm) covers all proper "vertices" of $\{p_1, \cdots, p_k\}$.

Now we are ready to state subprocedures to reduce the number of paths:

Procedure $CONCATENATION$ (G, P, M)
$G = (V, E)$ is a dense graph. $P = \{p_1, \cdots, p_k\}$ is a collection of disjoint paths, s.t. its union is V and $p_1, \cdots, p_\ell$ are proper. M is a maximal matching on $\tilde{H}$, where $\tilde{H}$ is defined as above.

Begin
For each endpoint x of any p_j and each p_i, s.t. $\{p_i, x\} \in M$ pick up an endpoint y of p_i, s.t. $[y, x] \in E$ and do (y, x) into F.

(**Comment:** Each vertex x has at most one y, s.t. $(y, x) \in F$ and at most one y, s.t. $(x, y) \in F$)
End of Comment.)

For each maximal sequence $(x_1, \cdots, x_p)$ (called F-*chain*), s.t. $(x_i, x_{i+n}) \in F$ for each $i = 1 \cdots p$ set (x_{2i-1}, x_{2j}) into F', where $j = 1 \cdots \lceil \frac{p}{2} \rceil$.

Comments:
i) At most one vertex of any F-chain is not in any directed F'-edge.
ii) For each p_i, s.t. $[p_i, x] \in M$ and $x \in V$, one endpoint is in an F-chain (length ≥ 2).
iii) Therefore: At most $\frac{1}{3}$ of all vertices appearing in an F-chain do not appear in any directed edge of F'.
End of Comments.

For each $(x_1, x_2) \in F'$ concatenate the paths p_i, p_j belonging to x_1, x_2 resp. by the edge $[x_1, x_2]$.

(**Comment:** $\frac{2}{3}$ of the paths p_i, s.t. there is an $x, [p_i, x] \in M$ are concatenated.

Let P_1 be the set of all paths p, s.t. $[p_1, x] \in M$ and $x \in V$
Then the number of paths is reduced by at least $\frac{1}{3}|P_1|$
End of Comment.)

Procedure $INSERT$ (G, P, M)
declaration as in $CONCATENATION$

Begin
For each $p_i, i = 1 \cdots \ell$, s.t. $[p_i, [y_1, y_2]] \in M$ and $[y_1, y_2] \in p_j$, $(p_i \notin P_1)$ do $(p_i, p_j) \in \tilde{F}_1$. For each $\tilde{F}_1$-cycle delete one $(x_1, x_2) \in \tilde{F}_1$, which is on the cycle.

Comment: $\tilde{F}_1$ is now a directed forest and each $p_i \notin P_1$ belongs to one of the trees of $\tilde{F}_1$ of cardinality of at least 2.
End of Comment.

For each p_i, s.t. $(p_i, [y_1, y_2]) \in M$ and $(p_i, p_j) \in \tilde{F}_1$ insert p_i into $[y_1, y_2]$.

(**Comment:** Let $P_2 := \{p_i | i = 1 \cdots \ell, p_i \notin P_1\}$. Then the number of paths is reduced by at least $\frac{1}{2}|P_2|$).
End of Comment.)

Therefore

Lemma 8. Let $P = \{p_1 \cdots p_k\}$ be a set of disjoint paths covering all vertices and ℓ be the number of its proper paths. Then the concatenation of the procedures $CONCATENATION$ and $INSERT$ reduces the number of paths by at least $\frac{1}{3}\ell$.

We will now deal with improper paths (which are extensible to cycles). We assume that we have at most $\frac{n}{\lfloor 4 \rfloor}$ disjoint paths. Each (also improper) path p of size $|p|$ has at least $\frac{n}{2} - |p| + 1$ vertices not in p adjacent to some vertex of p. In case $|p| \leq \frac{n}{4}$ there are at least as many vertices as $|p|$ which are adjacent to some vertex of p, but not in p. *Therefore any maximal matching M on $(P \dot\cup V, \tilde{E} = \{(p,v) | [v,x] \in E$ for some $x \in p$ and $v \notin p\})$ covers all paths of size $\leq \frac{n}{4}$.*

We consider the following mapping from the set P_i of improper paths to P. $f(p) =$ the path p of P, s.t. $v \in p$ and $(p, v) \in M$. Each repeated application of f ends in an f-cycle or in a path p, not in P_i or in a path p of size $> \frac{n}{4}$. We make f into a directed forest by making f undefined for one element of each f-cycle. Each f-tree has cardinality ≥ 2. We decompose f into trees of depth 1 in the following way:

Mark all f-leaves by 1. Mark $f(x)$ by 0 if x is marked by 1 and mark y by 1 if all x, s.t. $f(x) = y$ are marked by 0. (Such marking procedures can be done in NC by an alternating logspace machine with polynomial tree size (see [Ru])). For each x which is marked by 0, let T_x consist of all y, s.t. $f(y) = x$ and y is marked by 1. Then T_x is a tree of depth 1. All T_x are disjoint and only roots of any f-tree do not belong to any T_x. This procedure is called $DECOM(f)$. The value of $DECOM(f)$ is the set of all these T_x. For each T_x we can now apply the following reduction procedure:

Procedure $REDUCE$ (T_x, M)

Begin

make all leaves of T_x to cycles; for each leaf p of T_x select an edge $[u_p, v_p]$, s.t. $[p, v_p] \in M$ (M is the matching defined as above); each p is made into a path again with u_p as its last vertex; make the root r of T_x into a cycle if r is improper; orient $r = (v_1, \cdots, v_\mu)$.

Let $i_p =$ the i, s.t. $v_p = v_i$ for each leaf p of T_x. Sort all T_x leaves p in ascending order of i_p. Denote the j^{th} leaf of T_x by p_j and set $i_j := i_{p_j}$. If r is a cycle, set for each leaf p_j in parallel $p'_j := p_j \frown (v_{i_j}, v_{i_j+1(\bmod \mu)} \cdots v_{i_{j+1(\bmod |T_x|-1)}-1})$

If r is a proper path, set $p'_j := p_j \frown (v_{i_j}, v_{i_j+1}, \cdots, v_{i_{j+1}})$; $p'_0 := (v_0, \cdots, v_{i_1-1})$, if r is a proper path.

Begin block marking

If p'_j is a proper path, then $M(p_j) = 0$;
If p'_j is improper and j maximal, then $M(p'_j) = 1$ (if r is a proper path);
If p'_j is improper:
If $M(p'_{j+1(\bmod \mu)}) = 1$, then $M(p'_j) = 0$ and if $M(p'_{j+1(\bmod \mu)}) = 0$ then $M(p'_j) = 1$.

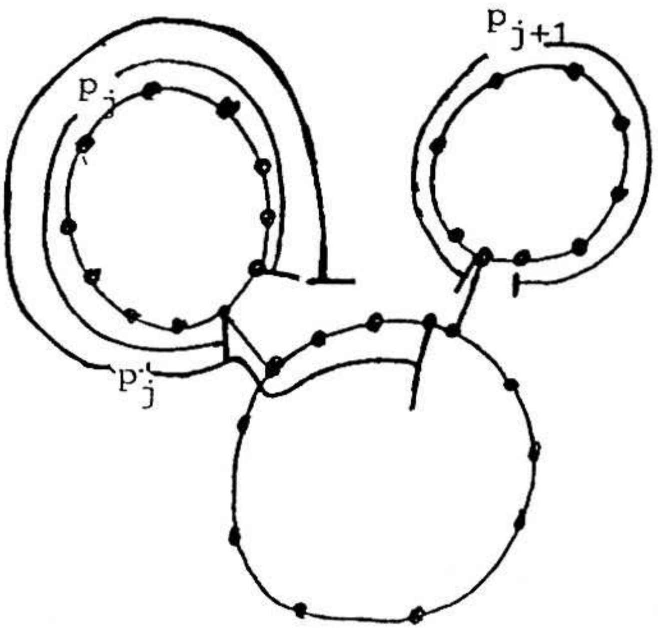

Figure 3: The construction of the paths p'_j in T_x

end block

(The block can be done in NC because it is in logspace).

For each j:

If $M(p'_j) = 1$

then begin

make p'_j a cycle; make v_{i_j} the endpoint of $p'_j; p'_j := p'_j \frown p'_{j-1(\bmod |T_x|-1)}$; delete $p'_{j-1(\bmod |T_x|-1)}$

end.

$REDUCE\ (T_x, M) := \{p'_j : p'_j$ not deleted $\}$.

end.

Let all p'_j be improper. Then $REDUCE\ (T_x, M)$ consists of $\frac{|T_x|}{\lfloor 2 \rfloor}$ paths.

In case r is a cycle, it consists of $\frac{|T_x|-1}{\lfloor 2 \rfloor}$ paths.

Let ℓ be the number of improper p'_j and m be the number of proper p'_j. Let ℓ_2 be the number of improper paths marked by 1. Then $\ell - \ell_1 \leq \ell_1$ and $REDUCE\ (T_x, M)$ consists of $m + \ell - \ell_1 \leq m + \frac{\ell}{2}$ paths, if r is a cycle; otherwise it consists of $\leq m + \frac{\ell}{2} + 1$ paths.

Therefore

Lemma 9. If $REDUCE\ (T_x, M)$ has m proper paths, then it has at most $\frac{|T_x|-m}{\lfloor 2 \rfloor}$ improper paths. In the case that the root r of T_x is improper, $REDUCE\ (T_x, M)$ consists of $\leq \frac{T_x - m - 1}{\lfloor 2 \rfloor}$ improper paths.

Now we state a procedure which reduces the number of paths by a factor:

Procedure $RD(G = (V, E), P)$ G is dense, P is a collection of disjoint paths covering V.

Begin

$Q := \{p \in P | p \text{ is improper }\}$;
$Q_1 := \{p \in Q | |p| \leq \frac{|v|}{4}\}$.

Compute by MIS-technique a maximal matching M_1 on $(Q_1 \dot{\cup} V, \{(p,v) | v \text{ is in the neighbourhood of } p\})$; Compute $f_1 : Q_1 \rightarrow P$, s.t. $(p,v) \in M \Rightarrow v \in f(p)$; $F_1 := DECOM(f_1)$; $R := \{p | p \notin \bigcup_{T \in F_1} T, p \in P\}$

$Q_2 := \bigcup_{T \in F_1} REDUCE(T, M_1) \cup R$

$P_1 := \{p \in Q_2 | p \text{ is proper }\}$: Let $\tilde{H} := (P_1 \dot{\cup} V', E)$ where $V' := \{v \in V | v \text{ is an endpoint of endpoint of some } p \in Q_2\} \cup \{e | e \text{ is some edge of any } p \in Q_2\}$, and $E := \{(p,v) | v \text{ is adjacent to an endpoint of } p \text{ and } v \in V'\} \cup \{(p_1, \{v_1, v_2\}) : \text{the one endpoint of } p \text{ is adjacent to } v_1 \text{ and the other endpoint of } p \text{ is adjacent to } v_2\}$.

Compute a maximal matching M_2 on $\tilde{H}$ by MIS. RD is the set of paths generated by concatenation of $CONCATENATION(G, Q_2, M_2)$ and $INSERT(G, Q_2, M_2)$.

end.

Analysis of *RD*.

Consider f_1 as a graph, s.t. the vertex set is the union of its domain and its range and the edge set is defined canonically. Each f_1-connected component has a cardinality ≥ 2. In the case that an f_1-component has cardinality $= 2$, both elements form one tree in $DECOM(f_1)$. Therefore at most $\frac{1}{3}$ of the improper paths $< \frac{n}{4}$ are not in any T of $DECOM(f_1)$.

Let n_i be the number of improper and n_p be the number of proper paths of P. Let m_i be the number of improper and m_p be the number of proper paths in any T of $DECOM(f_1)$. Let m'_p be the number of preper paths in $T \cup REDUCE(T, M_1)$. Let $k \leq 3$ be the number of improper paths $\geq \frac{1}{4}n$ not in any T of $DECOM(f_1)$.

Set $\mu := \frac{m_i}{n_i - k} \geq \frac{2}{3}$. Then Q_2 has at most $(1-\mu)(n_i - k) + \frac{\mu(n_i - k) + m_p - m'_p}{2} + 1$ improper paths. There remain $n_p - m_p + m'_p$ proper paths in Q_2. Here after the application of $CONCATENATION$ and $INSERT$ there remains at most the following number of paths: $(1-\mu)(n_i - k) + \frac{\mu(n_i - k) + m_p - m'_p}{2} + \frac{2}{3}(n_p - m_p + m'_p) = (n_i - k)(1 - \frac{\mu}{2}) + \frac{2}{3}n_p + \frac{1}{6}(m'_p - m_p) + 1 + k \leq \frac{2}{3}m_i - \frac{2}{3}k + \frac{2}{3}n_p + \frac{1}{6}(m'_p - m_p) + 1 + k \leq \frac{2}{3}(n_i + n_p) + \frac{1}{6}(m'_p - m_p) + \frac{k}{3} + 1 \leq \frac{2}{3}|p < + \frac{1}{6}|p| + 2 \leq \frac{5}{6}|p| + 2$.

That means:

Proposition 1: After $O(\log n)$ applications of RD one gets a number of paths bounded by 13.

Continuation of the algorithm:

The bounded number of paths allows us to work sequentially, not violating the NC-property:

Procedure $BP(P)$

1) Pick up a proper path $p \in P$ if it exists; otherwise any $p \in P$;
2) Insert p into another $q \in P$ or concatenate p with any other path q of P, if p is proper;
3) If p is improper, pick up an edge joining p to another $q \in P$. Make p and q cycles and concatenate them to a path.

Repeat BP 13 times. Then one has exactly one path. Clearly this path is improper. That means this path can be made into a cycle. Thus we have constructed a Hamiltonian cycle.

Time and Processor Analysis.

1) The first preparation of a collection of paths of length 2 or 3 needs time $O(\log^4 n)$ and $O(|E|^2) \leq O(n^4)$ processors.
2) Determining whether a path is proper or not needs $O(\log n)$ time and $O(n)$ processors. Therefore to determine it for all paths, we need $O(\log n)$ time and $O(n^2)$ processors.
3) To determine a matching preparing the procedures $CONCATENATION$ and $INSERT$, we need again $O(\log^4 n)$ time and $O(n^4)$ processors.
4) For the time and processor analysis of $CONCATENATION$, we first have to consider the computation of F-chains: This is a connected component and needs $O(\log n)$ time and $O(n^2)$ processors (see [Hi]). Then we have to divide the sets into the even and odd points of each F-chain (constructing F'). This can also be done in $O(\log n)$ time by $O(n)$ processors.

   To check which paths are concatenated, we need again a time of $O(\log n)$ and $O(n^2)$ processors. To check which paths are concatenated is a connected component problem and needs a time of $O(\log n)$ and $O(n)$ processors.

   The new paths need new addresses. This can be done by giving each of them the smallest address of the paths from which it is built up. This can be done in $O(\log n)$ time by $O(n)$ processors.

   Therefore:

   $CONCATENATION$ needs $O(\log n)$ time and $O(n^2)$ processors.
5) To check complexity of the procedure $INSERT$, we have to discover $\tilde{F}_1$-cycles: This can be done in $O(\log n)$ time by $O(n^2)$ processors. The insertion itself is a division of paths followed by a concatenation of paths which can be done in $O(\log n)$ time by $O(n^2)$ processors.
6) The maximal matching preparing $DECOM$ needs $O(\log^4 n)$ time by $O(n^4)$ processors.
7) Making f into a forest needs again $O(n^2)$ processors and $O(\log n)$ time.
8) $DECOM(f)$ is an alternating machine with logspace and treesize n. It needs $O(\log^2 n)$ time and $O(n^2)$ processors.
9) The parallel application of the procedure $REDUCE$ for all T_x is the reorganization and concatenation of paths which needs $O(n^2)$ processors and $O(\log n)$ time.
10) Therefore RD needs $O(\log^4 n)$ time and $O(n^4)$ processors.
11) The procedure BP needs $O(\log n)$ time and $O(n^2)$ processors.

Therefore

Theorem 3. For any dense graph one can determine a Hamiltonian cycle in time $O(\log^5 n)$ by $O(n^4)$ processors on a CREW-PRAM.

Remark: Using Luby's algorithm for MIS in the matching procedure, we can also get an algorithm using only $O(\log^3 n)$ time on a CREW-PRAM.

5. A Remark on Routing on Dense Graphs.

Dense graphs have an expander property (see[PU1], [PU2]) of high degree. Edge routing (that means connecting a set of pairs of vertices by edge disjoint paths) can be done trivially in AC^1 ([Co]), because any pair of vertices has a distance of at most two. Therefore any collection of shortest paths is edge disjoint.

An interesting problem is the *vertex routing* on dense graphs, which is the connection of a given set of pairs of vertices by the vertex disjoint paths.

Proposition 2: There is an infinite family of dense graphs with three pairs of vertices, s.t. vertex routing is not possible.

PROOF. We consider two cliques C_n, C'_n of vertex cardinality n and two additional vertices v_1, v_2, s.t. v_1 and v_2 are joined by an edge to all vertices C_n and C'_n. Let x_1, x_2 and $x_3 \in C_n$ and $y_1, y_2, y_3 \in C'_n$. Then only two of the pairs x_i, y_i can be routed, because each path from x_i to y_i must pass v_1 and v_2. It can easily be checked that the graph constructed as above is dense. □

The reason that routing three pairs is not possible is that the graph is only two-connected.

We would like to prove the following statement: Let G be a k-connected dense graph and $(x_1, y_1), \cdots, (x_k, y_k)$ be pairs of vertices of G. Then there is a vertex disjoint collection of paths connecting each x_i by y_i, and *these paths can be constructed in NC*. But we can prove the following result:

Proposition 2: The construction of a routing of k paths on k-connected dense graphs is as hard as the construction of a bipartite perfect matching.

PROOF. Consider any bipartite graph $(U \dot\cup V, E)$, s.t. U and V have the same size k'. Let U_1, U_2, U_3, U_4 be copies of U and for $u \in U$ let u_i be the corresponding element of U_i. Let V_1 and V_2 be copies of V and let v_1, v_2 be defined analogously for $v \in V$.

Construct a dense graph $\hat{G} := (\hat{V}, \hat{E})$ as follows: $U_1 \cup U_2$ and $U_3 \cup U_4$ form a clique. V_1 and V_2 form a clique. For $\{u, v\} \in E$, $\{u_i, v_j\} \in \hat{E}$. For $\{u, v\} \notin E$ let $\{u_1, v_2\}, \{u_2, v_2\} \in \hat{E}, \{u_3, v_1\}, \{u_4, v_1\} \in \hat{E}$.
$\hat{G}$ is dense, $\hat{G}$ is $2k'$ connected iff G has a perfect matching. $\hat{G}$ has a routing between all (u_1, u_3) and (u_2, u_4) iff G has a perfect matching. There is a canonical $1-1-$ correspondence between the routings on $\hat{G}$ and the pairs $(u_1, u_3), (u_2, u_4)$ and the perfect matchings on $G' := (U \cup U' \dot\cup V \cup V', E')$ where U' and V' are copies of U and V, respectively, and $(u, v) \in E'$ iff $(u, v) \in E, (u', v) \in E'$ iff $(u, v) \in E$, $(u, v') \in E'$ iff $(u, v) \in E$, and $(u', v') \in E'$ iff $(u, v) \in E$.
□

We consider at last $(\frac{1}{2} + \epsilon)$-dense graphs, that means, each vertex has degree of at least $(\frac{1}{2} + \epsilon)n$. Then the intersection of neighbourhoods of any two vertices is at least $2(\frac{1}{2} + \epsilon \cdot n + 2 - n = n + 2\epsilon n + 2 - n = 2\epsilon n + 2$. Then for any collection of pairs of vertices of size $2\epsilon n + 2$ we can find a vertex disjoint collection of paths of length at most 2 by maximal matching:

1) Connect all pairs x_i, y_i, s.t. $\{x_i, y_i\} \in E$ by an edge and delete them;
2) For all remaining pairs x_i, y_i say $\{(x_i, y_i), z\} \in \hat{E}$ iff $\{x_i, z\}, \{z, y_i\} \in E$. (Each (x_i, y_i) has an $\hat{E}$-degree of at least $2\epsilon n + 2$).

3) Compute a maximal matching M on $\hat{E}$ (which is also a maximum matching) and select for $(x_i, y_i), z \in M$ the path (x_i, z, y_i). By the same argument as in proposition 2 we can find $(\frac{1}{2}+\epsilon)$ dense graphs and $2\epsilon n+3$ pairs of vertices which cannot be routed.

6. Conclusions.

The sequential deterministic algorithm computing a Hamiltonian cycle for any dense graph seems to be related to the probabilistic solution of Angluin and Valiant [AV], which computes for any graph with high probability a Hamiltonian cycle. It might have seemed possible to transform Frieze's [Fr] probabilistic parallel algorithm into a deterministic one. But we were not successful in dividing any dense graph into two dense graphs of nearly equal size by an NC-algorithm.

Acknowledgements.

Dominic Welsh originally drew our attention to the fast parallel computation problems on dense graphs by making us acquainted with Edward's results [Ed]. We are grateful to Mark Goldberg and Christos Papadimitriou for posing the problem of constructing a Hamiltonian cycle for dense graphs in parallel. We would also like to thank Ephraim Korach for some hints on the literature, and Richard Karp, Eli Upfal, and Avi Wigderson for many interesting conversations.

References.

[Ai] Aigner, M., *Graphentheorie*, Teubner Studienbücher Mathematik, Stuttgart 1983

[AV] Angluin, D., and Valiant, L., *Fast Probabilistic Algorithms for Hamiltonian Circuits and Matchings*, Proc. 9^{th} ACM STOC (1977), pp. 30-41

[Bo] Bollobas, B., *Extremal Graph Theory*, Academic Press, London 1978

[Br] Broder, A.Z., *How hard Is it to Marry at Random*, Proc. 18^{th} ACM STOC (1986), pp. 50-58

[Co] Cook, S., *A Taxonomy of Problems with Fast Parallel Algorithms*, Information and Control 64 (1985), pp. 2-22

[Di] Dirac, G.A., *Some Theorems on Abstract Graphs*, Proc. London Math. Soc., Ser. 3,2 (1952), pp. 69-81

[DK1] Dahlhaus, E., and Karpinski, M., *The Matching Problem for Strongly Chordal Graphs Is in NC*, Research Report No. 855-CS, University of Bonn 1986

[DK2] Dahlhaus, E., and Karpinski, M., *Perfect Matching for Regular Graphs is AC^0-Hard for the General Matching Problem*, Research Report No. 858-CS, University of Bonn 1986

[Fr] Frieze, A.M., *Parallel Algorithms for Finding Hamiltonian Cycles in Random Graphs*, Information Processing Letters 25 (1987), pp. 111-117

[Gi] Gibbons, A., *Algorithmic Graph Theory*, Cambridge University Press, Cambridge 1985

[GJ] Garey, M., and Johnson, D., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman & Company, San Francisco 1979

[GK] Grigoriev, D.Yu., and Karpinski, M., *The Matching Problem for Bipartite Graphs with Polynomially Bounded Permanents Is in NC*, Proc. 28^{th} IEEE FOCS (1987), pp. 166-172

[Go] Goldschlager, L., *Synchroneous Parallel Computation*, Journal of the ACM 29 (1982), pp. 1073-1086

[GS] Goldberg, M., and Spencer, T., *A New Parallel Algorithm for the Maximal Independent Set Problem*, Proc. 28^{th} IEEE FOCS (1987), pp. 161-165

[HM] Hembold, D., and Mayr, E., *Two-Processor Scheduling Is in NC*; in: *VLSI-Algorithms and Architectures*, Makedon et. al. ed., pp. 12-25

[KUW] Karp, R., Upfal, E., and Wigderson, A., *Constructing a Perfect Matching Is in Random NC*, Proc. 17^{th} ACM STOC (1985—, pp. 22-32

[KUW2] Karp, R., Upfal, E., and Wigderson, A., *Are Search and Decision Problems Computationally Equivalent*, Proc. 17^{th} ACM STOC (1985—, pp. 464-475

[KW] Karp, R., and Wigderson, A., *A Fast Parallel Algorithm for the Maximal Independent Set Problem*, Proc. 15^{th} ACM STOC (1984—, pp. 266-272

[KL] Karpinski, M., and Lingas, A., *Subtree Isomorphisms and Bipartite Matchings Are Mutually NC-Reducible*, Research Report No. 856-CS, University of Bonn 1986

[KVV] Kozen, D., Vazirani, U., and Vazirani, V., *NC-Algorithms for Comparability graphs, Interval Graphs, and Testing for Unique Perfect Matching*; in: *5^{th} Conference on Foundations of Software, Technical and Computer Science*, S. Maheshvari ed., LNCS 206, Springer-Verlag, Berlin 1985, pp. 496-503

[LLRS] Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., and Shmoys, D.B. (ed.), *The Traveling Salesman Problem*, Wiley & Sons, Chichester 1985

[LPV] Lev, G., Pippenger, M., and Valiant, L., *A Fast Parallel Algorithm for Routing in Permutation Networks*, IEEE-Transactions on Computers C-30 (1981), pp. 93-100

[LP] Lovász, L., and Plummer, M., *Matching Theory*, Mathematical Studies, Annals of Discrete Mathematics Vol. 25, North Holland, Amsterdam 1986

[Lu] Luby, M., *A Simple Parallel Algorithm for the Maximal Independent Set Problem*, Proc. 17^{th} ACM STOC (1985), pp. 1-9

[MVV] Mulmuley, K., Vazirani, U., and Vazirani, V., *Matching Is as Easy as Matrix Inversion*, Proc. 19^{th} ACM STOC (1987), pp. 345-354

[PS] Papadimitriou, C.H., Steiglitz, K., *Combinatoric Optimization: Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, N.J. 1982

[PU1] Peleg, D., and Upfal, E., *The Token Distribution Problem*, Proc. 2^{th} IEEE FOCS (1986), pp. 418-427

[PU2] Peleg, D., and Upfal, E., *Constructing Disjoint Paths on Expander Graphs*, Proc. 19^{th} ACM STOC (1987), pp. 264-273

[Ru] Ruzzo, W.L., *Tree Size Bounded Alternation*, JCSS 21 (1980), pp. 218-235

[So] Soroker, D., *Fast Parallel Algorithms for Finding Hamiltonian Paths and Cycles in a Tournament*, Report No. UCB/CSD 87/309, University of California, Berkeley 1986

[SS] Shamir, E., and Schuster, A., *Parallel Routing in Networks: Back to Circuit Switching?*, Manuskript, 1986