Canonical Labeling of Regular Graphs in Linear Average Time

Luděk Kučera

Department of Computer Science
University of Chicago
Chicago, Illinois

Department of Applied Mathematics
Charles University
Prague, Czechoslovakia

ABSTRACT

An algorithm is presented to compute a canonical form of regular graphs. There is a constant c such that for each constant d the average running time of the algorithm over all d-regular graphs with N vertices is not greater than cNd, provided N is sufficiently large.

1. INTRODUCTION

A canonical labeling is an algorithm which labels vertices of an input graph by distinct labels in such a way that graphs G_1 and G_2 are isomorphic if and only if the unique label preserving bijection of their vertex sets is an isomorphism. It is clear that the problem of finding a canonical labeling is at least as difficult as isomorphism testing of graphs.

The computational complexity of both problems is one of the most challenging open questions in the theory of computation as we only know that the problems belong to the class NP, but neither NP-completeness nor polynomial time solvability nor even a membership in NP $\cap$ coNP have been proved. At the present time, the fastest known algorithm for isomorphism testing of graphs runs in $O(\exp(c\sqrt{N \log N}))$ time [2] and the problem is considered to be computationally intractable. On the other hand canonical labeling of d-regular graphs for d fixed (and, more generally, labeling of graphs of bounded degree) can be done in polynomially bounded worst case time, see [13], [4].

However, the degree of the polynomial bound grows with the degree of vertices of investigated regular graphs and even in the simplest case of 3-regular graphs the fastest known algorithm runs in $O(N^3 \log N)$ time, see [11]. Moreover, known algorithms with guaranteed polynomial worst case time bounds are based on deep results of the group theory which makes their logical structure rather involved. Therefore a simple and fast heuristic algorithm which could be used in most cases would be of practical use.

The evidence that such an algorithm might exist was given in [3], where it was proved that a simple heuristic algorithm for canonical labeling of graphs runs in linear expected time, provided all graphs with a given number of vertices are equaly likely to occur as an input. However, the isomorphism testing of d-regular graphs is much more complicated.

First, probabilities of edges in random regular graphs are not independent, which complicates the analysis of their properties. Until very recently, the only method available has been based on results of Bollobás [5]. In the present paper, an alternative approach is used in the case of random regular graphs subjected to certain conditions (e.g. non-existence of short

0272-5428/87/0000/0271$01.00 © 1987 IEEE

cycles).

Moreover, random regular graphs are very sparse, which implies e.g. that wecan not rely on the fact that graphs with non-trivial automorphism groups are extremely unlikely. Methods of [12], further developped in the present paper, show how to overcome this difficulty by characterising all permutation groups of random d-regular graphs,which occur with probability greater than $p(1/N)$ for some polynomial p.

Finally, degrees of all vertices of regular graphs are equal and therefore more complicated invariants are necessary to distinguish between vertices. Initial segments of length $(0.5+\varepsilon)$ log N of vertex degree sequences, defined by Bollobás [6], give a way how to compute a canonical labeling of almost all d-regular graphs in time $O(N^{1.5+\varepsilon})$. Since initial segments of vertex degree sequences of the length less than $(0.5+\varepsilon)\log N$ are likely to be the same for most vertices, constant or even $O(N^{0.5+\varepsilon})$time per vertex is not sufficient to get a good partition of vertices using vertex degree sequences. To overcome this problem, our algorithm first labels vertices according their distances from the set of all vertices covered by cycles of the shortest length, and then uses this initial labeling to refine information included in vertex degree sequences (i-th item of the refined sequence of a vertex v describes how many vertices in distance i from v have certain label instead of just giving the number of vertices in the distance i from v). It can be proved that constant length initial segments of refined degree sequences uniquely characterize a large number of vertices and therefore

Theorem 1. There is an algorithm A_o for computing canonical form of graphs such that there exists a constant c such that for each constants d,Δ
there exists N_o such that
for each N greater than N_o
if A_o is applied to a graph chosen at random among all d-regular graphs with N vertices then
the probability that A_o fails to give a result is $O(N^{-\Delta})$ and
the expected running time is less or equal to cNd.

The probability of failure of A_o is so small that it is sufficient to compute canonical form of graphs on which A_o failed by arbitrary polynomial worst case time algorithm to prove

Theorem 2. There are an algorithm A for computing canonical form of graphs, a constant c and a function $n=n(d)$ such that A computes a canonical form of a graph selected at random among all d-regular graphs with N vertices in expected time at most cNd, provided $N \geq n(d)$.

2. DEFINITIONS AND LEMMAS

Throughout the paper,N denotes the number of vertices of investigated graph. We are going to study properties of graphs for $N \to \infty$. $\{1,\ldots,N\}$will be denoted by X d will denote the degree of vertices of regular graphs we are going to study. d is supposed to be a constant greater or equal to 3 and not to depend on N.All logarithms in the paper are of the base d-1. Given a set A, $|A|$ denotes the number of elements of A .

A relation E on X is a set of two-element subsets of X.Elements of E are called edges, elements of X are vertices.

$\deg(x,E)$ denotes the number of edges of E containing a vertex x. $REG_d(X)$ is the class of all relations F on X such that $\deg(x,F)=d$ for each $x \in X$. $\rho_E(x,y)$ is the length of the shortest path from x to y in E, $\rho_E(x,A) = \min\{\rho_E(x,a) : a \in A\}$.

Given a relation E containing a cycle,
$U_L(v,E)$ denotes the set of all vertices w such that $\rho_E(v,w) \leq L$,
$M_o(E)$ denotes the set of all vertices covered by shortest cycles of E,
$M_i(E)$ denotes the set of all vertices v such that $\rho_E(v,M_o(E))=i$,
$V(v,i,E)$, where $v \in M_j(E)$, $j \leq i$, is the set of all $w \in M_i(E)$ such that $\rho_E(w,v)=i-j$,
$m_i(E)=|M_i(E)|$, $n_i(E)=N-(m_o(E)+\ldots+m_i(E))$,
$D_E(v,i,j)$ is the number of vertices w such that $\rho_E(v,w)=i$ and $\rho_E(v,M_o(E))=j$,
$D_E(v,i) = D_E(v,i,0)+\ldots+D_E(v,i,N)$.
The sequence $(D_E(v,0),\ldots,D_E(v,k))$ is called (E,k)-sequence of the vertex v.

A vertex $v \in M_i(E)$, $i > 0$, is called E-proper if there are exactly d-1 vertices $v \in M_{i+1}$ such that $\{v,w\} \in E$, otherwise it is called E-improper.

If there is no danger of misunderstanding, the letter E will often be omitted.

Given $E \subset F \in REG_d(X)$, we write $E \lhd F$ if no edge of F-E is contained in any shortest cycle of F. A relation E is called extendible, if $E \lhd F$ for some $F \in REG_d(X)$.
Given a nonempty set A, a uniform random variable on A is the random variable $\overline{V}$ with values in A such that
$Prob(\overline{V} = a) = 1/|A|$ for each $a \in A$.
The expected value of a random variable $\overline{V}$ is denoted by $E(\overline{V})$.

The uniform random variable on $REG_d(X)$ will be denoted by $\overline{F}$. Given an extendible relation E, $\overline{F}_E$ denotes the uniform random variable over the set of all $F \in REG_d(X)$ such that $E \lhd F$.

Two inequalities are used throughout the paper. The first one is well known Chebychev inequality, see e.g. [7], the second one follows immediately from Chernoff bound [9]:

Lemma 2.1. Given a random variable $\overline{V}$ with values 0,1,2,...,
$Prob(|\overline{V} - E(\overline{V})| \geq t) \leq (E(\overline{V})^2 - E^2(\overline{V}))/t^2$,
especially

$$Prob(\overline{V}=0) \leq (E(\overline{V}^2) - E^2(\overline{V}))/E^2(\overline{V}) .$$

Lemma 2.2. Given a constant $\alpha > 0$, thereexists a constant $c=c(\alpha) > 0$ such that

$$\sum_{|k-np| > \alpha np} \binom{n}{k} p^k (1-p)^{n-k} < \exp(-cnp) .$$

3. CYCLES AND DEGREE SEQUENCES

Next two theorems will be used in analysis of Algorithm A_o

Theorem 3.1. All vertices covered by shortest cycles of $\overline{F}$ can be found in linear expected time and there are constants $\varepsilon, \delta > 0$ such that the probability that more than $N^{0.5-\delta}$ vertices are contained in cycles shorter than $\varepsilon \log N$ is $O(N^{-(0.5+\delta)})$.
Sketch of proof: All cycles of the length k in d-regular graph can be found in time $O(N \min(N,(d-1)^{k/2}))$. Bollobás proved that, for fixed k, the number of k-cycles in a random d-regular graph has assymptotically Poisson distribution with the mean $(d-1)^k/2k$ and this fact is essentially true even for slowly increasing k.

Theorem 3.2. [12] Given constants $\Delta, \varepsilon > 0$, there is a constant R such that, with probability $1-O(N^{-\Delta})$, all but R vertices have unique $(\overline{F}, \lfloor(0.5+\varepsilon) \log N\rfloor)$-sequence. Moreover, if $\Delta=1$ then we can choose R=0.

4. THE ALGORITHM

The algorithm A_o is based on subroutines REFINE and FINISH. During the computation, vertices of a graph are labeled by integers, and both procedures are called to refine the labeling.

Given a set A of vertices, REFINE(A) changes a labeling of elements of A so that two vertices $u,v \in A$ receive the same label iff they have had the same labels

and the sets (with repetition) of labels of their neighbours have been equal.

The procedure FINISH(A) is similar, but vertices u,v get the same labels if $D(u,i,j)=D(v,i,j)$ for all $i,j=1,\ldots,N$.

Theorem 3.2 says that, with probability $1-O(1/N)$, no two vertices of A have the same label after FINISH(A).

It is clear that $O(|A|d)$ time is sufficient to REFINE a set A. In order to FINISH a set A, it is usually unnecessary to compute all numbers $D(a,i,j)$. It is convenient to evaluate numbers $D(a,i,j)$ succesively for $i=1,2,\ldots$ usin breadth-first search, to avoid computation of those $D(a,i,j)$ which are equal to 0 by taking into account only vertices which are in distance i from the vertex a, and to interupt evaluation of $D(a,i,j),\ldots$ whenever $D(a,k,j)$, $k<i$, give the unique collection of invariants distinguishing the vertex a from all other vertices of the graph.

Thus, in view of Theorem 3.2, the probability that running time of FINISH(A) is $O(|A|N^{0.5+\varepsilon})$ is $1-O(1/N)$. In the last part of the analysis of the algorithm we use the fact that, under certain circumstances, the numbers $D(a,i,j)$, $i \leq c \log N$ are sufficient to partiton A into singletons. In such a case, FINISH(A) runs in $O(|A|N^{c})$ time.

NUQ denotes the set of all vertices of the graph, which are not uniquely labeled. Initial value of NUQ is computed in line 7 of the algorithm A_1 and then updated after any change of labeling of vertices by the procedures REFINE and FINISH (though operations with NUQ are not explicitely stated in the description of A_1).

It follows from the theorem 3.2, that with probability $1-O(1/N)$, FINISH(X) gives a complete partition of vertices in time $O(N^{0.5+\varepsilon})$ and almost complete partition (up to a finite number of vertices) with probability $1-O(N^{-4})$. In order to obtain a linear average time algorithm, it is sufficient to have an algorithm which runs in linear time with probability $1-O(N^{-(0.5+\varepsilon)})$ for some constant $\varepsilon>0$.

Throughout the rest of paper, $0<\varepsilon<1/4$ is a constantsuch that the probability that more than $N^{0.5-\varepsilon}$ vertices is covered by cycles of length less than $3+4\varepsilon \log N$ is $O(N^{-(0.5+\varepsilon)})$. The existence of such a constant follows from Theorem 3.1 (and the analysis of the proof shows that it is sufficient to choose $\varepsilon=1/30$ for $\delta=1/10$).

Algorithm A_1:

```
1.  begin
2.  L:=the length of the shortest cycle;
3.  M_o:=the set of all vertices covered by
        cycles of the length L;
4.  for all vertices v do label(v):=ρ(v,M_o);
5.  r:=max(label(v) : v∈X);
6.  for i:=1 to r do M_i:={v: label(v)=i};
7.  NUQ:=the union of all M_i s.t. |M_i| > 1;
8.  FINISH(M_o);
9.  for i:=1 to r do if |M_i| ≤ N^(0.5-2) then
    FINISH(M_i);
10. for i:=0 to r-1 do
11.   begin
12.   for j:=i+1 to min(r, ⌊i+ c log N⌋) do
      PHASE(i,j);
13.   FINISH(M_{i+1} ∩ NUQ);
14.   if M_{i+1} ∩ NUQ ≠ ∅ then failure;
15.   end;
16. end;
```

where the procedure PHASE is defined as follows:

```
procedure PHASE(i,j);
1.  begin
2.  Z_j:= the set of all improper vertices
        of M_j ∩ NUQ;
3.  {W_ij:=Z_j};
4.  k:=j;
5.  loop
6.    Z_{k-1}:=the set of all neighbours of Z_k
          in M_{k-1} ∩ NUQ;
7.    {W_ij:=W_ij ∪ Z_{k-1}};
8.    if Z_{k-1} = ∅ then exit loop;
```

```
 9.   k:=k-1;
10. end of loop
11. m:=k;
12. fork:=m to j do REFINE(Z_k);
13. {W'_ij:=∅};
14. for k:=j downto m do
15.   begin
16.   REFINE(Z_k);
17.   {W'_ij:=W'_ij ∪ (Z_k ∩ NUQ) ;
18.   FINISH(Z_k);
19.   end;
20. end;
```

The sets W_{ij} and W'_{ij} are not used in the computation, but they will be referred to in the proof of properties of the algorithm.

If A_1 doesn't fail then each vertex has the unique label and the labeling is isomorphism invariant. A canonical ordering of vertices can be obtained by sorting vertices according labels. Even if the set NUQ is nonempty but small at the and of computation, we can obtain a canonical form of the graph using a brute force method, e.g. finding all orderings π of vertices such that the sequence
label(π(i)), i=1,...,N,
is nondecreasing(there are at most (|NUQ|)! such orderings) computing all sequences
$A_\pi = (a_\pi(1),\ldots,a_\pi(dN))$
of elements of the set $\{1,\ldots,N\}$, where
$a_\pi(id-(d-1)) < \ldots < a_\pi(id-1) < a_\pi(id)$
for each i=1,...,N and
$\{\pi(i), a_\pi(id-j)\}$
is an edge for each i=1,...,N, j=0,...,d-1. (Note that A_π is uniquely determined by π, as it is a sequence of ordered lists of neighbours of vertices $\pi(1),\pi(2),\ldots,\pi(N)$. Then the ordering π which has the lexicographically smallest A_π is chosen as a canonical ordering of a graph. Using the radix sort (see e.g. [1]), the brute force computation requires time O(Nd(|NUQ|)!).

```
Algorithm A_o;
begin
if the input graph is connected then
   apply the algorithm A_1;
FINISH(X);
if (|NUQ|)! ≤ √N then
   compute the canonical form of G using
   the brute force method;
else failure;
end;

Algorithm A;
begin
apply the algorithm A_o;
if A_o failed then
   compute the canonical form of G using
   an algorithm with guaranted polynomial
   worst case running time;
end;
```

5. ANALYSIS OF ALGORITHM

We are going to study a behavior of the algorithm A_1 applied to $F \in REG_d(X)$ such that

(1) no two different vertices of X have the same $(F, \lfloor(0.5+\varepsilon)\log N\rfloor)$-sequence,
(2) the number of vertices covered by cycles shorter than $2+4\varepsilon \log N$ is at most $N^{0.5-\varepsilon}$,
(3) F is connected and the diameter of F is O(log N).

Theorems 3.1 and 3.2 imply thet the proposition (1) holds with probability 1-O(1/N) and (2) is satisfied with probability $O(N^{0.5-\varepsilon})$. The diamater of a random regular graph is almost surely O(log N) and, a random regular graph is almost surely connected, see[7]. Note that if (1) is fulfiled then FINISH(U) needs $O(|U|N^{0.5+\varepsilon})$ steps and no two different vertices of U have the same label after FINISH(U).

Lemma 5.1. Assuming (1) and (3), the algorithm doesn't fail and gives a complete partition of the set of vertices.

Proof: The algorithm executes FINISH($M_i \cap$ NUQ) for each i=0,...,r, see lines 8,13.

Lemma 5.2. Assuming (1) and (3), the expected running time of the algorithm is O(N) plus time necessary to perform all executions of line 18 of PHASE and lines 8 and 13 of the main program.

Proof: Lines 2,3 need O(N) expected time, see theorem 3.1. Using a breadth-first search (see e.g. [1]), we can finish line4 in O(N) steps. Lines 5,6 clearly need O(N) time. Supposing (1), all members of W_{ij} are deleted from NUQ during the execution of PHASE(i,j) (line 18) and therefore sets W_{ij} are disjoint for different pairs (i,j) Since one execution of lines 2-16 of PHASE(i,j) needs time $O(|W_{ij}|)$ and the sum of $|W_{ij}|$ over all couples i,j is at most N, the lemma follows.

Lemma 5.3. Assuming (1) and (2), the expected time necessary to perform line 8 of the algorithm and all executions of line18 of PHASE is O(N). Moreover, if $v \in M_{i+1}$ belongs to NUQ after execution of PHASE(i,j) then no direct pathfrom v to a vertex of M_j contains an improper vertex.(A path $v_1,\dots,v_k$ is called direct if there is i such that $v_j \in M_{i+j}$ for j=1,...,k).

Proof: (1) and (2) imply that line 8 needs linear time. Denote by C(k) the set of vertices contained in cycles of length at most k. Let us suppose that (1) and (2) are fulfiled. Sets W'_{ij} are pairwise disjoint and therefore, in view of (1), the time to be estimated is bounded by the sum of sizes of all of these sets, multiplied by $O(N^{0.5+\varepsilon})$. It is sufficient to prove that all sets W'_{ij} are included in $C(2+4\varepsilon \log N)$.

Let v be an improper vertex of M_i at beginning of execution of PHASE(i,j). Consider direct paths

$u_p,\dots,u_j$ and $v_q,\dots,v_j=v$
such that $u_p, v_q \notin$ NUQ,
$u_{p+1},\dots,u_j, v_{q+1},\dots,v_j \in$ NUQ
at the beginning of PHASE(i,j) and
either $u_j=v$, $u_{j-1} \neq v_{j-1}$
or $u_j \in M_j$ and u_j is connected with v.

Since M_i and NUQ are disjoint after FINISH($M_i \cap$ NUQ), which preceeds beginning of PHASE(i,j), such paths exist and p and q are greater or equal to i.

If $u_k=v_k$ for some k,then $v \in C(1+2\varepsilon \log N)$ Let us suppose the converse. Consider the statement in line 12 of PHASE.Since u_p, v_q do not belong to NUQ,which means that they are uniquely labeled, all vertices $w \in W_{ij}$ which are labeled by the same label as u_j (v_j, resp.) after execution of line 12 have to be connected with u_p (v_q,resp.) by a direct path. It follows that if $z \neq v$ is a vertex of M_j labeled by the same label as v at the beginning of FINISH(Z_j) in line 18 of PHASE(i,j) then z is connected with v_q by a direct path and equal or adjacent to a vertex of M_j, which is connected to u_p by a direct path. In such a case, $v \in C(2+4\varepsilon \log N)$.

Now, let $u,v \in Z_k$, $i<k<j$, be two different vertices of M_k, which have the same labels after execution of REFINE(Z_k) in line 16 of PHASE(i,j). There are direct paths

$u_p,\dots,u_k=u,\dots,u_j$ and $v_q,\dots,v_k=v,\dots,v_j$
such that $u_p, v_q \notin$ NUQ,
$u_{p+1},\dots,u_j, v_{q+1},\dots,v_j \in$ NUQ
at the beginning of PHASE(i,j) and u_j, v_j are improper. It follows from (1) that after execution of FINISH(Z_{k+1}) in line 18 of PHASE the vertices u_{k+1} and v_{k+1} have the same labels if and only if they are equal. IF REFINE(Z_k) doesn't distinguish u and v than u_{k+1} must be a neighbour of v

In the same way as above, we can prove that u and v are not distinguished in line 12 of PHASE only if there is a direct path from u_p to v. Hence, if $u,v \in Z_k$ are not distinguished before the execution of FINISH(Z_k), both of them belong to the set $C(2\varepsilon \log N)$.

6. ANALYSIS OF ALGORITHM (cont'd)

The only part of the computation of A_1 which is not covered by the estimation given in Paragraph 5 is line 13. To prove that the global time spent in line 13 is linear on average, we need some lemmas.

Lemma 6.1. Let U,V be subsets of X, L be a natural number, E be an extendible ralation. Define

$$a=\sum_{u\in U} f(u), \quad b=\sum_{v\in V} f(v), \quad n=\sum_{x\in X} f(x),$$

where $f(x)=d-deg(x,E)$.

Let p_k be the probability that $\overline{F}_E$ contains exactly k edges $\{u,v\}$ such that $u\in U$, $v\in V$, $\{u,v\}\notin E$ and p'_k be tha probability of the same event, but conditional on the assumption that neither U nor V contain an edge of $\overline{F}_E - E$. Then

$$\frac{p_k}{p_{k-1}} \leq \frac{ab}{k(n-2(d-1)^{L+1}-2(a+b)}\,,$$

$$\frac{p'_k}{p_{k-1}} = \frac{(a-k+1)(b-k+1)}{k(n-2(a+b-k))}\,(1+h) \text{ for } U\cap V=\emptyset,$$

where

$$h = O\left(\frac{(d-1)^L}{b-k}\right) + O\left(\frac{(d-1)^L}{n-2(a+b-k)}\right)$$

and L is the length of the shortest cycle of E.

Proof: Choose a number r and put

$S_k=\{F\in REG_d(X)$: $E\lhd F$ and there are exactly k edges $\{u,v\}\in F-E$ such that $u\in U$, $v\in V\}$,

$S'_k=\{F\in S_k$: neither U nor V contain an edge of $F-E\}$.

T_k (T'_k, resp.) be the set of all 5-tuples (F,u,v,w,z), where $F\in S_k$ ($F\in S'_k$, resp.), $u\in U$, $v\in V$, $w,z\notin U\cup V$, $\{u,v\},\{w,z\}\in F$, $F\cup\{\{u,w\},\{v,z\}\}-\{\{u,v\},\{w,z\}\}\in S_{k-1}$, ($S'_{k-1}$, resp.).

Given $F\in S_k$, we are going to estimate the number of 4-tuples (u,v,w,z) such that $(F,u,v,w,z)\in T_k$. There are at least k and at most 2k (exactly k if U,V are disjoint sets) choices of an ordered pair (u,v) and from n-2(a+b) to n possible choices of an oriented couple (w,z) such that $w,z\notin U\cup V$, $\{w,z\}\in F$. (The number of choices is exactly n-2(a+b-k) if U,V are disjoint and $U\cup V$ contains k edges of F-E). On the other hand if an edge $\{u,v\}\in F-E$, $u\in U$, $v\in V$ is chosen arbitrarily and $\{w,z\}\in F-E$ then $(F,u,v,w,z)\in T_k$, provided

(a) $w,z\notin U\cup V$,

(b) there is no edge of F connecting $\{u,v\}$ and $\{w,z\}$
(conditions (a) and (b) imply that $\{u,w\},\{v,z\}\notin F$, $F\cup\{\{u,w\},\{v,z\}\}-\{\{u,v\},\{w,z\}\}\in REG_d(X)$, $E\subset F$, $F\cup\{\{u,w\},\{v,z\}\}-\{\{u,v\},\{w,z\}\}$ contains k-1 edges connecting U,V),

(c) there is no path of length less than L connection $\{u,v\}$ and $\{w,z\}$
((c) implies that no edge of $F\cup\{\{u,w\},\{v,z\}\}-\{\{u,v\},\{w,z\}\}-E$ is contained in a cycle of F of the length at most L, because $E\lhd F$ implies that only $\{u,w\}$ and $\{v,z\}$ coulb be such adges.)

At most $2(d-1)^{L+1}$ edges do not satisfy (c) (note that the number of choices of u, v,w,z such that $(F\cup\{\{u,w\},\{v,z\}\}-\{\{u,v\},\{w,z\}\})\in T_k$ is

between $k(n-2(a+b)-2(d-1)^{L+1})$ and 2kn in general case and

between $k(n-2(a+b-k)-2(d-1)^{L+1})$ and $k(n-2(a+b-k))$ if U,V are disjoint and $U\cup V$ contains exactly k edges of F-E.

Now, given $F \in S_{k-1}$, let us estimate the number of 4-tuples (u,v,w,z) such that $F \cup \{\{u,v\},\{w,z\}\} - \{\{u,w\},\{v,z\}\} \in T_k$. $\{u,w\}, \{v,z\}$ have to be edges of F-E such that $u \in U$, $v \in V$, $w,z \notin U \cup V$. There are at most ab possibilities how to choose such edges and if U,V are disjoint and neither U nor V contain an edge of F-E then there are exactly a-(k-1) (b-(k-1), resp.) edges of F-E intersecting both U (V,resp.) and X-(U ∪ V).

Among such edges, it is sufficient to choose u,w arbitrarily and thenselect $\{v,z\}$ so that $\{u,w\}$ and $\{v,z\}$ do not interset and there is no path of length less than L, connecting $\{u,w\}$ and $\{v,z\}$. Only at most $2(d-1)^{L+1}$ vertices v,z do not satisfy the later condition.

It follows that
$|S_k| k\ (n-2(a+b)-2(d-1)^{L+1}) \leq |T_k| \leq |S_{k-1}|\, ab$,
if U and V are disjoint then
$|S'_k| k\ (n-2(a+b-k)-2(d-1)^{L+1}) \leq |T'_k|$,
$|T'_k| \leq |S_{k-1}|\ (a-k+1)(b-k+1)$,
$|S'_k| k\ (n-2(a+b-k)) \leq |T'_k|$,
$|T'_k| \leq |S'_{k-1}|\ (a-k+1)(b-k+1-2(d-1)^{L+1})$.

Lemma 6.1 says that, if a,b and $(d-1)^L$ are sufficiently small, then p_k, (p'_k,resp.) is negligible, unless k is less or equal (equal, resp.) to
$(1+o(1))\ ab/n$.
Based on this, we are able to prove

Theorem 6.2 Algorithm A_1 runs in linear expacted time.

Sketch of the proof: Put $j = i + \lfloor \varepsilon \log N \rfloor$ and denote by T_i the value of $M_i \cap NUQ$ at the beginning of FINISH($M_i \cap NUQ$)in line 13 of the algorithm A_1. Lemma 5.3 implies that V(u,k) contains an improper vertex for no $u \in T_i$, $i \leq k \leq j$.

The proof that FINISH(T_i) needs $O(N/\log N)$ expected time is divided into several cases:

if $|T_i| << N^{0.5}$ (especially if $m_i << N^{0.5}$ or $r < j$) then it is sufficient to apply the condition (1).

If $N^{\varepsilon} m_j/n_j \geq N^{-\beta}$ for some small constant $\beta > 0$ then, given $u \in M_i$ and putting $U=V(u,j)$ and $V=M_j$, the lemma 6.1 implies that it is very unlikely that U contains no improper vertices and therefore T_i is almost surely very small or even empty.

If $N^{\varepsilon} m_j/n_j = N^{-\beta}$ then for $\mu = 2\beta/3$
$N^{\mu+\varepsilon} m_j/n_j = N^{-\beta/3}$ and $N^{2\mu+\varepsilon} m_j/n_j = N^{\beta/3}$.
Zhe first inequality implies that $m_j N^{\mu+\varepsilon} << N$ and therefore $(F, \lfloor (\varepsilon+\mu)\log N \rfloor)$-sequences of all vertices of T_i can be found in sublinear time.

Moreover, if $k = j + \mu \log N$ then
$m_q/n_q << 1$ for $q = j,\ldots,k$
and Lemma 6.1 implies that the "density" of improper vertices in M_q is of order m_q/n_q and therefore small and hence the respective sizes of V(u,k), $u \in T_i$ and M_k are roughly $N^{(\varepsilon+\mu)}$ and $m_j N^{\mu}$.

In virtue of Lemma 6.1, we can expect many improper vertices of V(u,k) for $u \in T_i$ (roughly $N^{\mu+\varepsilon} m_j N^{\mu}/n_k \geq N^{2\mu+\varepsilon} m_j/n_j = N^{\beta/3}$). Moreover, in view of small density of improper vertices, the expected size of the intersection of sets V(u,k) and V(v,k) is very small with respect to the sizes of V(u,k) and V(v,k) for any two different elements $u,v \in T_i$, because $V(u,j) \cap V(v,j) = \emptyset$ and we would need too many cycles to obtain large $V(u,q) \cap V(v,q)$ for larger $q \leq k$.

Let $\bar{g}_q(u)$ denotes the number of improper vertices of V(u,q). Given two different vertices $u,v \in T_i$ and a number q close to k, $\bar{g}_q(u)$ and $\bar{g}_q(v)$ are two almost independent random variables with large expectations and therefore the probability that they have exactly the same values is very small. This proves that the numbers of improper vertices in V(u,q) and V(v,q) are likely to be different. Since the numbers D(u,q,j), j=1,...,N, reflect the number of improper vertices in V(u,q) we can suppose with large probability that numbers
D(u,q,j), $u \in T_i$, q k, j=1,...,N
give a partition of the set T_i into singletons.

Theorem 6.3. Algorithm A_o runs in linear average time and almost surely does not fail.

Algorithm A runs in linear average time and never fails.

Proof: A_1 runs in $o(N)$ average time and it fails with probability $O(N^{-(0.5+\beta)})$ for some constant $\beta > 0$, FINISH(X) runs in $O(N^2)$ worst case time with probability of failure $O(N^{-\Delta})$, where Δ is a constant such that there is a $O(N^{\Delta})$ worst case algorithm for computing canonical form of d-regular graphs.

Moreover, it follows from theorem 3.2, that the probability that FINISH finishs after $O(N^{0.5+\beta})$ steps is $O(1/N)$.

Thus, the average running time of A is

$$O(N)+O(N^{-(0.5+\beta)})O(N^{0.5+\beta})+O(N^{-1})O(N^2)+$$
$$+O(N^{-\Delta})O(N^{\Delta}) = O(N).$$

REFERENCES

1] A.V.Aho, J.E.Hopcroft, J.D.Ullman, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, Mass., 1974

2 L.Babai, W.M.Kantor, E.M.Luks, Computational complexity and classification of finite simple groups, 24th Annual Conference on Foundation of Computer Science, 1983, 162-171.

[3] L.Babai, L.Kučera, Canonical labelling of graphs in linear average time, 20th Annual Conference on Foundations of Computer Science, 1979, 39-46.

[4] L.Babai, E.M.Luks, Canonical labeling of graphs, 15th Symposium on Theory of computing, 1983, 171-183.

[5] B.Bollobás, A probabilistic proof of an asymptotic formula for the number of labelled regular graphs, Europ.J. Combinatorics, 1(1980), 311-316.

[6] B.Bollobás, Distinguishing vertices of random graphs, in Graph Theory, Annals Discrete Math. 13(1982), 33-50.

[7] B.Bollobás, Random Graphs, Academic Press, London 1985.

[8] B.Bollobás, V.Fernandez dela Vega, The diameter of random regular graphs, Combinatorica 2(1982), 125-134.

[9] H.Chernoff, A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations, Annals Math.Statist. 23(1952),493-509.

[10] M.Fürer, W.Schnyder, E.Specker, Normal forms for trivalent graphs and graphs of bounded valence, 15th Annual Symposium on Theory of Computing, 1983, 161-170.

[11] Z.Galil, C.M.Hoffman, E.M.Luks, C.P. Schnorr, A.Weber, An $O(n^3 \log n)$ deterministic and $O(n^3)$ probabilistic test for isomorphism of trivalent graphs, 23rd Annual Conference of Foundation of Computer Science, 1982, 118-125.

[12] L.Kučera, An $O(n^{1.5+})$ expected time algorithm for canonization and isomorphism testing of trivalent graphs, 2nd Symposium on Theoretical Aspects of Computer Science, Lecture Notes in Computer Science 182, 1985, 197-207.

[13] E.M.Luks, Isomorphism of bounded valence can be tested in polynomial time, 21st Annual Conference on Foundation of Computer Science, 1980, 42-49.

[14] N.C.Wormald, The asymptotic distribution of short cycles in random regular graphs, J.Combinatorial Theory, ser.B, 31 (1981), 162-182.