Graph Automorphism Computation

Robert L. Miller

February 2, 2008

Refining partitions

- An *equitable partition* of a labeled graph G is a partition $\pi = [V_0|V_1|...|V_k]$ of $V(G)$ such that

  $$d_G(v_1, V_2) = d_G(v_2, V_2) \text{ for all } v_1, v_2 \in V_1 \text{ and all } V_1, V_2 \in \pi.$$

Refining partitions

- An *equitable partition* of a labeled graph G is a partition $\pi = [V_0|V_1|...|V_k]$ of $V(G)$ such that

  $$d_G(v_1, V_2) = d_G(v_2, V_2) \text{ for all } v_1, v_2 \in V_1 \text{ and all } V_1, V_2 \in \pi.$$

- The refinement of a partition π of $V(G)$ by a list of subsets α, denoted $R(G, \pi, \alpha)$, was described last time, and is loosely given by the algorithm:

$R(G, \pi, \alpha)$

For each set W in α:
—For each set V in π:
——Split V up by its degree to W (in order), resulting in new cells $X_1, ..., X_s$,
——Update α as follows: if the current cell of π is a set in α, then replace that cell set in α with X_i for $|X_i|$ maximal, and put the other X_js at the end.

Partition Nests

- Given a labeled graph G, a partition π and a sequence of vertices $v_1, \ldots, v_{m-1}$, the *partition nest* determined by these is a sequence of partitions $(\pi_1, \ldots, \pi_m)$ defined by

a) $\pi_1 = R(G, \pi, \pi)$, and

b) $\pi_i = \pi_{i-1} \perp v_{i-1} := R(G, \pi_{i-1} \circ v_{i-1}, \{\{v_{i-1}\}\})$, where if $\pi_{i-1} = [V_1|\ldots|V_k]$ and $v_{i-1} \in V_j$, then

$$\pi_{i-1} \circ v_{i-1} := [V_1|\ldots|V_{j-1}|v_{i-1}|V_j \setminus \{v_{i-1}\}|V_{j+1}|\ldots|V_k].$$

Partition Nests

- Given a labeled graph G, a partition π and a sequence of vertices $v_1, ..., v_{m-1}$, the *partition nest* determined by these is a sequence of partitions $(\pi_1, ..., \pi_m)$ defined by

a) $\pi_1 = R(G, \pi, \pi)$, and

b) $\pi_i = \pi_{i-1} \perp v_{i-1} := R(G, \pi_{i-1} \circ v_{i-1}, \{\{v_{i-1}\}\})$, where if $\pi_{i-1} = [V_1|...|V_k]$ and $v_{i-1} \in V_j$, then

$$\pi_{i-1} \circ v_{i-1} := [V_1|...|V_{j-1}|v_{i-1}|V_j \setminus \{v_{i-1}\}|V_{j+1}|...|V_k].$$

- BDM usually uses Greek letters to represent partition nests, i.e. nodes of the tree. We especially use ν, ρ, ζ and η.

The Search Tree

- The *search tree* $T(G, \pi)$ is the set of all partition nests starting at π, where the tree structure is given by common partitions: $(\pi_1, ..., \pi_k)$ is a descendant of $(\pi_1, ..., \pi_{k-1})$, for example.

The Search Tree

- The *search tree* $T(G, \pi)$ is the set of all partition nests starting at π, where the tree structure is given by common partitions: $(\pi_1, ..., \pi_k)$ is a descendant of $(\pi_1, ..., \pi_{k-1})$, for example.
- A node $\nu = (\pi_1, ..., \pi_k)$ of the search tree is a *terminal node* iff π_k is the discrete partition. In this case, the ordering of the partition π_k defines an ordering of the vertices of G, i.e. a new labeled graph, where we take the ordering from π_k. Notation: $G(\nu)$.

The Search Tree

- The *search tree* $T(G, \pi)$ is the set of all partition nests starting at π, where the tree structure is given by common partitions: $(\pi_1, ..., \pi_k)$ is a descendant of $(\pi_1, ..., \pi_{k-1})$, for example.
- A node $\nu = (\pi_1, ..., \pi_k)$ of the search tree is a *terminal node* iff π_k is the discrete partition. In this case, the ordering of the partition π_k defines an ordering of the vertices of G, i.e. a new labeled graph, where we take the ordering from π_k. Notation: $G(\nu)$.
- More notation: if $\nu = (\pi_1, ..., \pi_k)$, then $\nu^{(i)} = (\pi_1, ..., \pi_i)$ for $i \leq k$.

The Search Tree

- The *search tree* $T(G, \pi)$ is the set of all partition nests starting at π, where the tree structure is given by common partitions: $(\pi_1, ..., \pi_k)$ is a descendant of $(\pi_1, ..., \pi_{k-1})$, for example.
- A node $\nu = (\pi_1, ..., \pi_k)$ of the search tree is a *terminal node* iff π_k is the discrete partition. In this case, the ordering of the partition π_k defines an ordering of the vertices of G, i.e. a new labeled graph, where we take the ordering from π_k. Notation: $G(\nu)$.
- More notation: if $\nu = (\pi_1, ..., \pi_k)$, then $\nu^{(i)} = (\pi_1, ..., \pi_i)$ for $i \leq k$.
- If two nodes ν_1, ν_2 are not descendants of each other, then for some i, we have $\nu_1^{(i)} = \nu_2^{(i)}$ but $\nu_1^{(i+1)} \neq \nu_2^{(i+1)}$. Define $\nu_1 - \nu_2 = \nu_1^{(i+1)}$.

More on the Search Tree

- If ν_1 is an ancestor of ν_2, then $\nu_1 < \nu_2$. Otherwise, there is a node $(\pi_1, ..., \pi_m)$ and vertices $v_1 \neq v_2$ such that

$$\nu_1 - \nu_2 = (\pi_1, ..., \pi_m, \pi_m \perp v_1) \text{ and}$$

$$\nu_2 - \nu_1 = (\pi_1, ..., \pi_m, \pi_m \perp v_2).$$

Then define $\nu_1 < \nu_2$ iff $v_1 < v_2$.

More on the Search Tree

- If ν_1 is an ancestor of ν_2, then $\nu_1 < \nu_2$. Otherwise, there is a node $(\pi_1, \dots, \pi_m)$ and vertices $v_1 \neq v_2$ such that

$$\nu_1 - \nu_2 = (\pi_1, \dots, \pi_m, \pi_m \perp v_1) \text{ and}$$

$$\nu_2 - \nu_1 = (\pi_1, \dots, \pi_m, \pi_m \perp v_2).$$

  Then define $\nu_1 < \nu_2$ iff $v_1 < v_2$.
- Suppose $\gamma \in S_n$ such that $G^\gamma = G$ and $\pi^\gamma = \pi$. If $\nu_1, \nu_2 \in T(G, \pi)$ and $\nu_1^\gamma = \nu_2$ for some such γ, we say $\nu_1 \sim \nu_2$. This defines an equivalence, and we say a node ν is an *identity node* if it is the earliest node in its equivalence class. Fact: if $\nu_1 < \nu_2$ and $\nu_1 \sim \nu_2$, then $T(G, \pi, \nu_2 - \nu_1)$ contains no identity nodes.

Linear Ordering of the Set of Labeled Graphs

- This is done by enumerating the set of labeled graphs as follows:

Linear Ordering of the Set of Labeled Graphs

- This is done by enumerating the set of labeled graphs as follows:

from sage.graphs.graph "one can" import enum

```
from sage.rings.integer import Integer
M = graph.am()
string = ''
for r in M.rows():
    for c in r:
        string += str(c)
if string=='': string='0'
return Integer(string,2)
```

Linear Ordering of the Set of Labeled Graphs

- This is done by enumerating the set of labeled graphs as follows:

from sage.graphs.graph "one can" import enum

```
from sage.rings.integer import Integer
M = graph.am()
string = ''
for r in M.rows():
    for c in r:
        string += str(c)
if string=='': string='0'
return Integer(string,2)
```

- Optimization!

The Indicator Function $\Lambda(G, \pi, \nu)$

- Given a labeled graph G, ordered partition of $V(G), \pi$ and partition nest $\nu \in T(G, \pi)$, we define an *indicator function* $\Lambda(G, \pi, \nu) \in \Delta$ which satisfies:
  - Δ has a linear ordering.
  - For any $\gamma \in S_n$, $\Lambda(G, \pi, \nu) = \Lambda(G^\gamma, \pi^\gamma, \nu^\gamma)$,

from sage.graphs.graph_isom "one can" import indicator

```
from sage.misc.misc import prod
LL = [0]*G.order()
for partition in V:
    a = len(partition)
    for k in range(a):
        LL[k] += len(partition[k])*(1 + sum(\
[  degree(G, partition[k][0], partition[i])\
   for i in range(len(partition))  ]  ) )
return prod([l for l in LL if l!=0])
```

The Canonical Label $C(G, \pi)$

- Given Λ, we can define another ordering $\tilde{\Lambda}$ by

$$\tilde{\Lambda}(G, \pi, \nu) := (\Lambda(G, \pi, \nu^{(1)}), \ldots, \Lambda(G, \pi, \nu^{(k)}))$$

  where $k = |\nu|$, and we use the lexicographic ordering induced by Δ.

The Canonical Label $C(G, \pi)$

- Given Λ, we can define another ordering $\tilde{\Lambda}$ by

$$\tilde{\Lambda}(G, \pi, \nu) := (\Lambda(G, \pi, \nu^{(1)}), \ldots, \Lambda(G, \pi, \nu^{(k)}))$$

  where $k = |\nu|$, and we use the lexicographic ordering induced by Δ.
- If $X(G, \pi)$ is the set of terminal nodes of $T(G, \pi)$, then we define the canonical label $C(G, \pi) := \max\{G(\nu) | \nu \in X(G, \pi) \text{ and } \tilde{\Lambda}(G, \pi, \nu) = \Lambda^*\}$, where $\Lambda^* = \max\{\tilde{\Lambda}(G, \pi, \nu) | \nu \in X(G, \pi)\}$.

The Canonical Label $C(G, \pi)$

- Given Λ, we can define another ordering $\tilde{\Lambda}$ by

$$\tilde{\Lambda}(G, \pi, \nu) := (\Lambda(G, \pi, \nu^{(1)}), \ldots, \Lambda(G, \pi, \nu^{(k)}))$$

  where $k = |\nu|$, and we use the lexicographic ordering induced by Δ.
- If $X(G, \pi)$ is the set of terminal nodes of $T(G, \pi)$, then we define the canonical label $C(G, \pi) := \max\{G(\nu) | \nu \in X(G, \pi) \text{ and } \tilde{\Lambda}(G, \pi, \nu) = \Lambda^*\}$, where $\Lambda^* = \max\{\tilde{\Lambda}(G, \pi, \nu) | \nu \in X(G, \pi)\}$.
- As noted last time, two graphs are isomorphic iff they have the same canonical label.

The Canonical Label $C(G, \pi)$

- Given Λ, we can define another ordering $\tilde{\Lambda}$ by

$$\tilde{\Lambda}(G, \pi, \nu) := (\Lambda(G, \pi, \nu^{(1)}), ..., \Lambda(G, \pi, \nu^{(k)}))$$

  where $k = |\nu|$, and we use the lexicographic ordering induced by Δ.
- If $X(G, \pi)$ is the set of terminal nodes of $T(G, \pi)$, then we define the canonical label $C(G, \pi) := \max\{G(\nu) | \nu \in X(G, \pi) \text{ and } \tilde{\Lambda}(G, \pi, \nu) = \Lambda^*\}$, where $\Lambda^* = \max\{\tilde{\Lambda}(G, \pi, \nu) | \nu \in X(G, \pi)\}$.
- As noted last time, two graphs are isomorphic iff they have the same canonical label.
- Define a *canonical node* to be a node ν such that $G(\nu) = C(G, \pi)$.

- **Lemma 2.18** If $\gamma \in S_n$ and ν is a terminal node, then $G(\nu^\gamma) = G(\nu)$ iff $\gamma \in \text{Aut}(G)_\pi$.

- **Lemma 2.18** If $\gamma \in S_n$ and ν is a terminal node, then $G(\nu^\gamma) = G(\nu)$ iff $\gamma \in \mathrm{Aut}(G)_\pi$.
- **Theorem 2.20** Suppose $X^*(G, \pi)$ is any subset of $X(G, \pi)$ which contains those identity nodes ν for which $\tilde{\Lambda}(G, \pi, \nu) = \Lambda^*$. Then $X^*(G, \pi)$ contains a canonical node.

Pruning the Tree I

- What we want to do in terms of Lemma 2.18 is to reduce the size of $X^*(G, \pi)$ as much as possible. We do this using automorphisms. Suppose we discover a ν_2 such that $G(\nu_2) = G(\nu_1)$ for some earlier ν_1, and both are terminal nodes (there is a $\gamma \in S_n$ such that $\nu_2 = \nu_1^\gamma$). Then Lemma 2.18 implies $\gamma \in \mathrm{Aut}(G)_\pi$. Call this an *explicit automorphism.* As mentioned before, we can now ignore the subtree $T(G, \pi, \nu_2 - \nu_1)$. However, if we have a bunch of these, we know that the subgroup they generate A is also in $\mathrm{Aut}(G)_\pi$. BDM takes advantage of this information as follows.

Pruning the Tree II

- Define ζ to be the earliest terminal node of $T(G, \pi)$. Then:

Pruning the Tree II

- Define ζ to be the earliest terminal node of $T(G, \pi)$. Then:
- **Lemma** If $\nu_1 < \nu_2$ and both are in $X(G, \pi)$, then $|\zeta - \nu_2| \leq |\nu_1 - \nu_2|$. (pf- otherwise $\nu_2 \in T(G, \pi, \zeta - \nu_1)$.)

Pruning the Tree II

- Define ζ to be the earliest terminal node of $T(G, \pi)$. Then:
- **Lemma** If $\nu_1 < \nu_2$ and both are in $X(G, \pi)$, then $|\zeta - \nu_2| \leq |\nu_1 - \nu_2|$. (pf- otherwise $\nu_2 \in T(G, \pi, \zeta - \nu_1)$.)
- Initialize θ as the discrete partition. Whenever we obtain an explicit automorphism γ, update θ by replacing it with the finest partition coarser than both θ and the orbit of γ. Thus θ is always the orbit partition of A, the subgroup of $\mathrm{Aut}(G)_\pi$ that we have so far. Also, θ is finer than the orbit partition of $\mathrm{Aut}(G)_{\pi_m}$ where $(\pi_1, ..., \pi_m)$ is any common ancestor of all terminal nodes so far considered (a permutation taking one node to another fixes their common ancestors).

Pruning the Tree III

- If $\nu = (\pi_1, ..., \pi_m)$ is an ancestor of ζ and of all the terminal nodes so far considered, then let $W = \{v_1, ..., v_k\}$ be the first smallest nontrivial cell of π_m, where the v_i are in order. Since θ is finer than π_m, it induces a partition of W. The successors of ν, in order, are $\nu(v_1), ..., \nu(v_k)$ where $\nu(v_i) = (\pi_1, ..., \pi_m, \pi_m \perp v_i)$. If $v_i < v_j$ are in the same cell of θ, there is some automorphism $\gamma \in A$ such that $\nu(v_j) = \nu(v_i)^\gamma$. Thus we can eliminate $T(G, \pi, \nu(v_j))$ from searching, by:
    - Consider only $T(G, \pi, \nu(v_i))$ for which v_i is a minimal cell representative of θ, and
    - upon discovering an explicit automorphism γ in generating $T(G, \pi, \nu(v_i))$, see if v_i is still a minimal cell representative of the updated θ. If not, then γ is proof that $T(G, \pi, \nu(v_i))$ only contains terminal nodes equivalent to those we have already considered.

Implicit Automorphisms

- **Lemma 2.25** Suppose G is a labeled graph and π an equitable partition. If π has m nontrivial cells and one of the following hold:

  a) $n \leq |\pi| + 4$,
  b) $n = |\pi| + m$, or
  c) $n = |\pi| + m + 1$,

  then π_1 is the orbit partition of $\mathrm{Aut}(G)_{\pi_1}$ for any equitable π_1 finer than π.

- **Lemma 2.25** Suppose G is a labeled graph and π an equitable partition. If π has m nontrivial cells and one of the following hold:
  a) $n \leq |\pi| + 4$,
  b) $n = |\pi| + m$, or
  c) $n = |\pi| + m + 1$,

  then π_1 is the orbit partition of $\mathrm{Aut}(G)_{\pi_1}$ for any equitable π_1 finer than π.
- Whenever we encounter a node $\nu = (\pi_1, ..., \pi_m)$ for which π_m satisfies the Lemma, all the terminal nodes descended from ν must be equivalent, so we need only check one.

Canonical Label Candidates

- We use the variable ρ to find the canonical label. It is initialized as $\rho := \zeta$, and every time we find a terminal node ν with either
  a) $\tilde{\Lambda}(G, \pi, \nu) > \tilde{\Lambda}(G, \pi, \rho)$, or
  b) $\tilde{\Lambda}(G, \pi, \nu) = \tilde{\Lambda}(G, \pi, \rho)$ and $G(\nu) > G(\rho)$,

  we update $\rho := \nu$.

- We use the variable ρ to find the canonical label. It is initialized as $\rho := \zeta$, and every time we find a terminal node ν with either
  a) $\tilde{\Lambda}(G, \pi, \nu) > \tilde{\Lambda}(G, \pi, \rho)$, or
  b) $\tilde{\Lambda}(G, \pi, \nu) = \tilde{\Lambda}(G, \pi, \rho)$ and $G(\nu) > G(\rho)$,

  we update $\rho := \nu$.
- Suppose $\rho = (\pi_1, ..., \pi_m)$ and $\nu = (\pi'_1, ..., \pi'_k)$ is not necessarily terminal. Let $r := \min\{m, k\}$. Then if $\tilde{\Lambda}(G, \pi, \nu^{(r)}) < \tilde{\Lambda}(G, \pi, \rho^{(r)})$, we know that (by definition of indicator function) $\tilde{\Lambda}(G, \pi, \nu') < \tilde{\Lambda}(G, \pi, \rho)$ for every terminal node ν' of $T(G, \pi, \nu)$.

Sketch of the Algorithm I

- Start by creating $\nu = \rho = \zeta$. Define $m := |\zeta|$, and $r := |\rho|$.

Sketch of the Algorithm I

- Start by creating $\nu = \rho = \zeta$. Define $m := |\zeta|$, and $r := |\rho|$.
- Suppose we have just created $\nu = \nu^{(k)}$. Denote $\tilde{\Lambda} := \tilde{\Lambda}(G, \pi, \nu)$.

- Start by creating $\nu = \rho = \zeta$. Define $m := |\zeta|$, and $r := |\rho|$.
- Suppose we have just created $\nu = \nu^{(k)}$. Denote $\tilde{\Lambda} := \tilde{\Lambda}(G, \pi, \nu)$.
  1. If ($k > m$ or $\tilde{\Lambda} \neq \tilde{\Lambda}(G, \pi, \zeta^{(k)})$) and ($k > r$ or $\tilde{\Lambda} < \tilde{\Lambda}(G, \pi, \rho^{(k)})$), go to B.
  2. If ν is nonterminal, search $T(G, \pi, \nu)$.
  3. If $k > m$ or $\tilde{\Lambda} \neq \tilde{\Lambda}(G, \pi, \zeta)$, go to 4.
     Else, if the permutation γ taking ζ to ν is an automorphism, go to A.
  4. If ($k > r$ or $\tilde{\Lambda} < \tilde{\Lambda}(G, \pi, \rho)$) or ($\tilde{\Lambda} = \tilde{\Lambda}(G, \pi, \rho)$ and $G(\nu) < G(\rho)$, go to B.
     If ($\tilde{\Lambda} > \tilde{\Lambda}(G, \pi, \rho)$) or ($\tilde{\Lambda} = \tilde{\Lambda}(G, \pi, \rho)$ and $G(\nu) > G(\rho)$), update $\rho := \nu$ and go to B.
     If $\tilde{\Lambda} = \tilde{\Lambda}(G, \pi, \rho)$ and $G(\nu) = G(\rho)$, define γ taking ρ to ν and go to A.

Sketch of the Algorithm II

A Here we have found an explicit automorphism.

- Update θ to be the finest partition coarser than θ and than the orbit partition of γ, and store information about γ.
- Let v be the vertex such that if the longest common ancestor of ζ and ν is $\nu^{(h)}$, $\pi_{h+1} = \pi_h \perp v$. If v is not a minimum cell representative of θ, then return to $\nu^{(h)}$. Otherwise, return to the longest common ancestor of ν and ρ.

Sketch of the Algorithm III

B Here we are considering a terminal node not known to be equivalent to any earlier terminal node.

- If π_k satisfies Lemma 2.25, define hh to be the smallest value of $i \leq k$ such that π_i also satisfies 2.25. Otherwise, $hh := k$.
- If $hh < k$, store information about π_{hh}.
- Return to $\nu^{(i)}$, where $i = \min\{hh - 1, \max\{ht - 1, hzb\}\}$,
    - Define ht to be the smallest $i \leq m$ for which all the terminal nodes descended from or equal to $\zeta^{(i)}$ have been shown to be equivalent.
    - Define hzb to be the largest $i \leq \min\{k, r\}$, such that $\tilde{\Lambda}(G, \pi, \nu^{(i)}) = \tilde{\Lambda}(G, \pi, \rho^{(i)})$.

But don't just take my word for it...

TRY IT!!!

- http://cs.anu.edu.au/~bdm/papers/pgi.pdf