Computing High Stringency COGs Using Turán Graphs

Craig Falls, UNC CS
Jack Snoeyink, UNC CS
Bradford Powell, UNC Genomics
Comparative Genome Studies

**Biological Terminology**
- sequenced genomes
- mutual best-fit pairs
- Clusters of Orthologous Groups (COGs)

**Mathematical Terminology**
- string of DNA base pairs
- graph
  \[ V = \text{genes} \]
  \[ E = \text{best-fit pairs} \]
- sets of cliques

**Flow Diagram**

1. sequenced genomes
2. mutual best-fit pairs
3. Clusters of Orthologous Groups (COGs)
4. BLAST
5. COG algorithm
6. biologists
7. function prediction
8. phylogenetic classification
9. ancestor studies
Comparative Genome Studies

**Biological Terminology**

- sequenced genomes
- mutual best-fit pairs
- Clusters of Orthologous Groups (COGs)

**Mathematical Terminology**

- string of DNA base pairs
- graph
  - $V = \text{genes}$
  - $E = \text{best-fit pairs}$
- sets of cliques

**Processes**

1. BLAST
2. COG algorithm
3. biologists
4. function prediction
5. phylogenetic classification
6. ancestor studies
Comparative Genome Studies

**Biological Terminology**

- sequenced genomes
- mutual best-fit pairs
- Clusters of Orthologous Groups (COGs)

**Mathematical Terminology**

- string of DNA base pairs
- graph
  - $V = $ genes
  - $E = $ best-fit pairs
- sets of cliques

**Diagram**

- BLAST
- COG algorithm
- biologists
- function prediction
- phylogenetic classification
- ancestor studies
**Comparative Genome Studies**

**Biological Terminology**
- sequenced genomes
- mutual best-fit pairs
- Clusters of Orthologous Groups (COGs)

**Mathematical Terminology**
- string of DNA base pairs
- graph $V = \text{genes}$
- $E = \text{best-fit pairs}$
- sets of cliques
- function prediction
- phylogenetic classification
- ancestor studies
Genomes $\xrightarrow{\text{BLAST}}$ **Mutual Best-Hit Graph**

![Diagram of the mutual best-hit graph](image)
COMPARATIVE GENOME STUDIES

<table>
<thead>
<tr>
<th>Biological Terminology</th>
<th>Mathematical Terminology</th>
</tr>
</thead>
<tbody>
<tr>
<td>sequenced genomes</td>
<td>string of DNA base pairs</td>
</tr>
<tr>
<td>mutual best-fit pairs</td>
<td>graph</td>
</tr>
<tr>
<td>Clusters of Orthologous Groups (COGs)</td>
<td>$V = \text{genes}$</td>
</tr>
<tr>
<td></td>
<td>$E = \text{best-fit pairs}$</td>
</tr>
<tr>
<td></td>
<td>sets of cliques</td>
</tr>
<tr>
<td></td>
<td>function prediction</td>
</tr>
<tr>
<td></td>
<td>phylogenetic classification</td>
</tr>
<tr>
<td></td>
<td>ancestor studies</td>
</tr>
</tbody>
</table>
Comparative Genome Studies

<table>
<thead>
<tr>
<th>Biological Terminology</th>
<th>Mathematical Terminology</th>
</tr>
</thead>
<tbody>
<tr>
<td>sequenced genomes</td>
<td>string of DNA base pairs</td>
</tr>
<tr>
<td>mutual best-fit pairs</td>
<td>graph</td>
</tr>
<tr>
<td>Clusters of Orthologous Groups (COGs)</td>
<td>$V =$ genes</td>
</tr>
<tr>
<td>biologists</td>
<td>$E =$ best-fit pairs</td>
</tr>
<tr>
<td>function prediction</td>
<td>sets of cliques</td>
</tr>
<tr>
<td>phylogenetic classification</td>
<td></td>
</tr>
<tr>
<td>ancestor studies</td>
<td></td>
</tr>
</tbody>
</table>
Comparative Genome Studies

**Biological Terminology**
- sequenced genomes
- mutual best-fit pairs
- Clusters of Orthologous Groups (COGs)

**Mathematical Terminology**
- string of DNA base pairs
- graph
  - \( V = \) genes
  - \( E = \) best-fit pairs
- sets of cliques

**Biologists**
- function prediction
- phylogenetic classification
- ancestor studies
A COG at stringency $m$ is the union of the vertices in a set of cliques of size $m$ connected by cliques of size $m-1$. [Tatusov '97]
A **COG** at stringency $m$ is the union of the vertices in a set of cliques of size $m$ connected by cliques of size $m-1$. [Tatusov '97]

**Example:**

*Level 2:*

- **gene**
- **mutual best hit**
- **COG**
COG Definition

A COG at stringency $m$ is the union of the vertices in a set of cliques of size $m$ connected by cliques of size $m-1$. [Tatusov '97]

Example:

Level 3:
A **COG** at stringency $m$ is the union of the vertices in a set of cliques of size $m$ connected by cliques of size $m-1$. [Tatusov '97]

**Example:**

**Level 4:**

- gene
- mutual best hit
- COG
COGs Form a Hierarchy
Computing COGs by Clique Enumeration

Example with stringency 3

Graph Algorithm COGs
**Computing COGs by Clique Enumeration**

Example with stringency 3

- **Gene**
- **Mutual best hit**
- **3 clique**
- **Shared 2 clique**
Computing COGs by Clique Enumeration

Example with stringency 3
Computing COGs by Clique Enumeration

Example with stringency 3
COMPUTING COGs BY CLIQUE ENUMERATION

Example with stringency 3

Problem: too many cliques to enumerate
New Algorithm vs Clique Enumeration

Motivation: improve performance for data sets of practical importance

Number of Vertices 4,606
Number of Edges 159,081
Clique Runtime 3 hr 58 min
New Algorithm Runtime 11 sec

How is a 4 order of magnitude improvement achieved?
**COG Computation is NP-hard**

**Maximum Clique Problem**

**Instance:** A graph $G = (V,E)$ and a positive integer $J < |V|$.

**Question:** Does $G$ contain a clique of size at least $J$?

Karp proved that this problem is NP-hard in '72.

**Theorem**

COG computation is **NP-hard** by reduction from the maximum clique problem.

**Proof (Reader's Digest version)**

There is at least one COG of stringency $m$ if and only if the input graph has an $m$-clique.
**NATURE OF THE INPUT**

**Algorithm**

**Graph**

**COGs**

**Type of gene relationship**

- **Orthologs**
  (genes shared through evolution)

- **Orthologs and Paralogs**
  (genes related by duplication events)

**Induced subgraph**

- **Clique**

- **Turán graph**
  (complete 3-partite graph)
**NATURE OF THE INPUT**

- **Type of gene relationship**
  - Orthologs (genes shared through evolution)
  - Orthologs and Paralogs (genes related by duplication events)

- **Induced subgraph**
  - Clique
  - Turán graph (complete 3-partite graph)

Input graphs can be described by fewer Turán graphs than cliques.
We call a complete $k$-partite graph a "Turán graph".

**Example:**

Turán (1941) characterized the graphs with $n$ vertices, no $(k+1)$-cliques, and the maximum number of edges as complete $k$-partite graphs with $\frac{n}{k}$ or $\frac{n}{k} + 1$ vertices in each partition.
Utilizing Turán Graphs

**Note:** A Turán graph spanning $m$ species is a set of $m$-cliques sharing $(m-1)$-cliques.

**Example**

In this Turán graph spanning 3 species,
Utilizing Turán Graphs

**Note:** A Turán graph spanning $m$ species is a set of $m$-cliques sharing $(m-1)$-cliques.

**Example**

this 3-clique

species 1  species 2  species 3
Utilizing Turán Graphs

**Note:** A Turán graph spanning $m$ species is a set of $m$-cliques sharing $(m-1)$-cliques.

**Example**

and this 3-clique
**Utilizing Turán Graphs**

**Note:** A Turán graph spanning $m$ species is a set of $m$-cliques sharing $(m-1)$-cliques.

**Example**

share this 2-clique.

species 1  species 2  species 3
**Utilizing Turán Graphs**

**Note:** A Turán graph spanning $m$ species is a set of $m$-cliques sharing $(m-1)$-cliques.

**Original COG Definition**
A COG at stringency $m$ is the union of the vertices in a set of cliques of size $m$ connected by cliques of size $m-1$. [Tatusov '97]

**Turán-based COG Definition**
A COG at stringency $m$ is the union of the vertices in a set of Turán graphs spanning $m$ vertices connected by vertices spanning $m-1$ species.
Definition
A maximal Turán graph is one which is not a proper subset of any Turán graph.
**COG Algorithm (Overview)**

**Input:** a graph  
-- vertices are genes  
-- edges are mutual best-hit pairs

**Algorithm phase 1:** find all maximal Turán graphs  
-- branch and bound  
-- avoid subsets of known maximal Turán graphs

**Intermediate structure:** Turán relation graph  
-- vertices are maximal Turán graphs  
-- weight of vertex is the number of species spanned  
-- weight of edge is the size of the largest shared clique

**Algorithm phase 2:** find connected components = COGs  
-- series of depth-first searches

**Output:** COGs at all stringencies
**Input:** a graph
-- vertices are genes
-- edges are mutual best-hit pairs
**COG Algorithm (Overview)**

**Input:** a graph
--- vertices are genes
--- edges are mutual best-hit pairs

**Algorithm phase 1:** find all maximal Turán graphs
--- branch and bound
--- avoid subsets of known maximal Turán graphs

**Intermediate structure:** Turán relation graph
--- vertices are maximal Turán graphs
--- weight of vertex is the number of species spanned
--- weight of edge is the size of the largest shared clique

**Algorithm phase 2:** find connected components = COGs
--- series of depth-first searches

**Output:** COGs at all stringencies
Turán Enumeration Pseudo-Code

Pseudo-code for phase 1

MaximalTurán ( G )
    for each edge (v1,v2) in G do
        max_Turán := Turán of vertex[v1] intersect Turán of vertex[v2]
        feasible := mutual neighbors of v1 and v2 in G
        cur_Turán := {v1,v2}
        Branch ( feasible, max_Turán, cur_Turán )

Branch ( feasible, max_Turán, cur_Turán )
    vertices_to_avoid := choose a vertex set from max_Turán
    new_feasible := feasible
    for each v in feasible
        if v in vertices_to_avoid then continue
        results := Bound (v, new_feasible, max_Turán, cur_Turán)
        if results != nil and vertices_to_avoid = nil
            then vertices_to_avoid := choose a vertex set from results
        max_Turán := max_Turán union results
        new_feasible := new_feasible - {v}
    return max_Turán

Bound ( vertex, feasible, max_Turán, cur_Turán )
    Turán of vertex := Turán of vertex[vertex]
    max_Turán := max_Turán intersect Turán of_vertex
    cur_Turán := {vertex} union cur_Turán
    feasible := feasible intersect neighborhoods of vertex in G
    if feasible = nil then
        if max_Turán = nil then
            for each vertex in cur_Turán
                Turán of vertex[vertex] := Turán of vertex[vertex] union cur_Turán
            return max_Turán
        else return nil
    else Branch ( feasible, max_Turán, cur_Turán )
Applying Known Maximal Clique Algorithms

Original graph

species 1
species 2
species 3

Intra-species edges added

species 1
species 2
species 3

- gene
- inter-species edge
- mutual best hit
- cross-species clique

Turán graph

mutual best hit
Applying Known Maximal Clique Algorithms

Original graph

- species 1
- species 2
- species 3

Intra-species edges added

- species 1
- species 2
- species 3

- gene
- inter-species edge
- mutual best hit
- cross-species clique

Turán graph
Applying Known Maximal Clique Algorithms

Original graph

Intra-species edges added

- gene
- mutual best hit
- Turán graph
- gene
- inter-species edge
- mutual best hit
- cross-species clique
Efficient Maximal Clique Algorithms

Algorithms for enumerating maximal cliques:

1) **Auguston-Minker '70**
   \[ O(m^2) \]

2) **Bron-Kerbosh '71**
   no known complexity bounds but empirically faster than (1)

3) **Tsukiyama '77**
   \[ O(|V| \cdot |E| \cdot m) \]
   (where \( m = \) number of maximal cliques)
**Theorem:** Phase 1 requires exponential time.

**Proof:** A graph can have an exponential number of maximal Turán graphs.

Consider graph $G=(V,E)$:

$V = \{v_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq n\}$

$E = \{(v_{ij}, v_{kl}) \mid i \neq k\}$

**Example (n=3):**

```
1       2       3
1 1 1 1 2 2 2
2 2 2 2 3 3 3
3 3 3 3 1 1 1
```
**Theorem:** Phase 1 requires exponential time.

**Proof:** A graph can have an exponential number of maximal Turán graphs

Consider graph $G=(V,E)$:

$V = v_{ij}$ for $1 \leq i \leq n$, $1 \leq j \leq n$

$E = (v_{i,j}, v_{k,l})$ for $i \neq k$, $j \neq l$

**Example (n=3):**

Every permutation of $\{1,2,3\}$ corresponds to a maximal Turán graph.

There is at least $n!$ maximal Turán graphs from only $n^2$ vertices.
**COG Algorithm (Overview)**

**Input:** a graph
   -- vertices are genes
   -- edges are mutual best-hit pairs

**Algorithm phase 1:** find all maximal Turán graphs
   -- branch and bound
   -- avoid subsets of known maximal Turán graphs

**Intermediate structure:** Turán relation graph
   -- vertices are maximal Turán graphs
   -- weight of vertex is the number of species spanned
   -- weight of edge is the size of the largest shared clique

**Algorithm phase 2:** find connected components = COGs
   -- series of depth-first searches

**Output:** COGs at all stringencies
Intermediate structure: Turán relation graph
-- vertices are maximal Turán graphs
-- weight of vertex is the number of species spanned
-- weight of edge is the size of the largest shared clique
Intermediate structure: Turán relation graph
-- vertices are maximal Turán graphs
-- weight of vertex is the number of species spanned
-- weight of edge is the size of the largest shared clique

Original graph
species 1 species 2 species 3

Turán relation graph

related Turáns

gene

mutual best hit

Turán graph
**Turán Relation Graph**

**Intermediate structure:** Turán relation graph
-- vertices are maximal Turán graphs
-- weight of vertex is the number of species spanned
-- weight of edge is the size of the largest shared clique

**Original graph**
species 1  species 2  species 3

**Turán relation graph**

- Related Turáns
- Gene
- Mutual best hit
- Turán graph
**COG Algorithm (Overview)**

**Input:** a graph
-- vertices are genes
-- edges are mutual best-hit pairs

**Algorithm phase 1:** find all maximal Turán graphs
-- branch and bound

**Intermediate structure:** Turán relation graph
-- vertices are maximal Turán graphs
-- weight of vertex is the number of species spanned
-- weight of edge is the size of the largest shared clique

**Algorithm phase 2:** find connected components = COGs
-- series of depth-first searches

**Output:** COGs at all stringencies
Algorithm Phase 2

Pseudo-code for phase 2:

for each v in V[G]
    for each pair mt1,mt2 of maximal Turán graphs in Turâns_of_vertex[v]
        key := <mt1,mt2>
        if Hashtable_has_key( key )
            then bitv := Hashtable_get( key )
        else bitv := bit vector with all 0s
        Hashtable_replace( key, bitv )
    return the hashtable

for each <mt1,mt2,bitv> in the hashtable
    edge := <mt1,mt2>
    edge_weight := number of 1s in bitv
    E := {(edge,edge_weight)} union E
    V := {mt1,mt2} union V
    return G = (V,E)

for each level 3 < m < |S| do
    V := vertices in G with weight >= m
    E := edges in G with weight >= m-1
    connected_components := DFS(V,E)
    for each component in connected_components do
        vertices_in_cog := nil
        for each maximal Turán t in component do
            vertices_in_cog := vertices_in_cog union {t}
        done
        report vertices_in_cog as next COG at level m
    done
done
**Algorithm Phase 2**

**Algorithm phase 2**: find connected components = COGs
- series of depth-first searches
- at stringency $m$: edge weight $\geq m-1$; vertex weight $> m$

**Turán relation graph**

**COGs at stringency 2**

- related Turáns
- gene
- mutual best hit
- Turán graph
Algorithm Phase 2

Algorithm phase 2: find connected components = COGs
-- series of depth-first searches
-- at stringency $m$: edge weight $\geq m-1$; vertex weight $> m$

Turán relation graph

COGs at stringency 3

related Turáns

- gene
- mutual best hit
- Turán graph
Algorithm phase 2: find connected components = COGs
-- series of depth-first searches
-- at stringency $m$: edge weight $\geq m-1$; vertex weight $> m$

Turán relation graph

COGs at stringency 4
Graph
Algorithm
COGs

**COG Algorithm (Overview)**

**Input:** a graph
  -- vertices are genes
  -- edges are mutual best-hit pairs

**Algorithm phase 1:** find all maximal Turán graphs
  -- branch and bound

**Intermediate structure:** Turán relation graph
  -- vertices are maximal Turán graphs
  -- weight of vertex is the number of species spanned
  -- weight of edge is the size of the largest shared clique

**Algorithm phase 2:** find connected components = COGs
  -- series of depth-first searches

**Output:** COGs at all stringencies
## Runtimes for New COG Algorithm

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Number of Edges</th>
<th>Number of Maximal Turans</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small</td>
<td>36,183</td>
<td>38,244</td>
<td>2 sec</td>
</tr>
<tr>
<td>Medium</td>
<td>297,551</td>
<td>535,196</td>
<td>46 sec</td>
</tr>
<tr>
<td>Large</td>
<td>900,056</td>
<td>1,527,406</td>
<td>53 sec</td>
</tr>
<tr>
<td>Huge</td>
<td>1,171,655</td>
<td>3,527,758</td>
<td>2 hr 58 min</td>
</tr>
<tr>
<td>NewHuge</td>
<td>1,729,907</td>
<td>4,077,530</td>
<td>2 hr 43 min</td>
</tr>
</tbody>
</table>

Turán Enumeration Performance for 5 data sets on 20 bacterial genomes
### Runtimes for Modified Input Shows Robustness

<table>
<thead>
<tr>
<th></th>
<th>Number of Edges</th>
<th>Number of Maximal Turans</th>
<th>Runtime (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Original</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>159,081</td>
<td>455,746</td>
<td>11</td>
</tr>
<tr>
<td><strong>Edges Added</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>+2.5%</td>
<td></td>
<td>468,960</td>
<td>12</td>
</tr>
<tr>
<td>+5%</td>
<td></td>
<td>480,714</td>
<td>14</td>
</tr>
<tr>
<td>+10%</td>
<td></td>
<td>506,078</td>
<td>18</td>
</tr>
<tr>
<td><strong>Edges Removed</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-2.5%</td>
<td></td>
<td>1,126,934</td>
<td>38</td>
</tr>
<tr>
<td>-5%</td>
<td></td>
<td>2,031,402</td>
<td>139</td>
</tr>
<tr>
<td>-10%</td>
<td></td>
<td>2,187,630</td>
<td>375</td>
</tr>
</tbody>
</table>
Numbers of COGs at Each Stringency

Numbers of COGs at stringency 3 through $m$ for 20 bacterial genomes.
Summary and Significance

Turán graphs allow COG computation to be orders of magnitude faster than straightforward clique enumeration.

<table>
<thead>
<tr>
<th>Number of Vertices</th>
<th>4,606</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Edges</td>
<td>159,081</td>
</tr>
<tr>
<td>Clique Runtime</td>
<td>3 hr 58 min</td>
</tr>
<tr>
<td>Turán Runtime</td>
<td>11 sec</td>
</tr>
</tbody>
</table>

We plan to integrate the algorithm into the COG database, which is currently limited to stringency 3 COGs.