Polynomial variants of the densest/heaviest k-subgraph problem

Maria Liazi[1]

joint work with
Ioannis Milis[2] Vassilis Zissimopoulos[1]

[1]Department of Informatics and Telecommunications

National and Kapodistrian University of Athens (NKUA)

[2]Department of Informatics
Athens University of Economics and Business (AUEB)

British **C**ombinatorial **C**onference, **2005**

- **Densest k-subgraph (DkS)**:
  Input: A graph $G = (V, E)$, $|V| = n$, and an integer k, $k \leq n$
  Output: Find a k vertices subgraph of G with the maximum number of edges

- **Heaviest k-subgraph (HkS)**:
  Input: A **weighted** graph $G = (V, E)$, $|V| = n$, and an integer k, $k \leq n$
  Output: Find a k vertices subgraph of G with the maximum total edge weight

- **Densest k-subgraph (DkS)**:
  Input: A graph $G = (V, E)$, $|V| = n$, and an integer k, $k \leq n$
  Output: Find a k vertices subgraph of G with the maximum number of edges

- **Heaviest k-subgraph (HkS)**:
  Input: A **weighted** graph $G = (V, E)$, $|V| = n$, and an integer k, $k \leq n$
  Output: Find a k vertices subgraph of G with the maximum total edge weight

- **DkS is strongly NP-hard** as generalization of the CLIQUE problem.

- **The problem remain strongly NP-hard even for:**
  - bipartite graphs, comparability graphs, chordal graphs [Corneil and Perl, DAM, 1984]
  - planar graphs [Keil and Brecht, JCMCC, 1991]
  - bipartite graphs even of maximal degree three regular graphs [Feige and Seltser, T.R. CS97-16, Weizmann Inst., 1997]

- **DkS is strongly NP-hard** as generalization of the CLIQUE problem.

- **The problem remain strongly NP-hard even for:**
  - bipartite graphs, comparability graphs, chordal graphs [Corneil and Perl, DAM, 1984]
  - planar graphs [Keil and Brecht, JCMCC, 1991]
  - bipartite graphs even of maximal degree three regular graphs [Feige and Seltser, T.R. CS97-16, Weizmann Inst., 1997]

- **Approximation algorithms**

  1. [Kortsarz and Peleg, FOCS, 1993]
  2. [Feige and Seltser, T.R. CS97 Weizmann Inst., 1997]
  3. [Srivastav and Wolf, APPROX, 1998; Erratum: TR U. Kiel 1999]
  4. [Ye and Zang, T.R. U. Iowa, 1999]
  5. [Asahiro et al., J. of Algorithms, 2000]
  6. [Feige et. al., Algorithmica, 2001]
  7. [Feige and Langberg, J. of Algorithms, 2001]
  8. [Han et al., Math. Progr., 2002]
  9. [Billionnet and Roupin, T.R. 486 Cedric-CNAM, 2004]

- **Best known approximation ratio for the DkS** $\rightarrow$ $O(n^{\frac{1}{3}})$, [6]
- **No one of them achieves a constant approximation ratio**

- **Approximation algorithms**

  1. [Kortsarz and Peleg, FOCS, 1993]
  2. [Feige and Seltser, T.R. CS97 Weizmann Inst., 1997]
  3. [Srivastav and Wolf, APPROX, 1998; Erratum: TR U. Kiel 1999]
  4. [Ye and Zang, T.R. U. Iowa, 1999]
  5. [Asahiro et al., J. of Algorithms, 2000]
  6. [Feige et. al., Algorithmica, 2001]
  7. [Feige and Langberg, J. of Algorithms, 2001]
  8. [Han et al., Math. Progr., 2002]
  9. [Billionnet and Roupin, T.R. 486 Cedric-CNAM, 2004]

- **Best known approximation ratio for the DkS** $\rightarrow$ $O(n^{\frac{1}{3}})$, [6]
- **No one of them achieves a constant approximation ratio**

- There is no a PTAS for the DkS problem [Khot, FOCS, 2004]
- There is not known inapproximability result for some approximation ratio
- There are constant approximation algorithms for special cases of the problems:
  - A PTAS for dense graphs [Arora et al., JCSS, 1999]:
    - of minimum degree $\Omega(n)$
    - of $\Omega(n^2)$ edges when k is $\Omega(n)$
  - Constant approximation algorithms for complete graphs where the weights satisfy the triangle inequality:
    - 4-approximation [Ravi et al., Oper. Res., 1994]
    - 2-approximation [Hassin et al., Oper. Res. Letters, 1997]

- There is no a PTAS for the DkS problem [Khot, FOCS, 2004]
- There is not known inapproximability result for some approximation ratio
- There are constant approximation algorithms for special cases of the problems:
  - A PTAS for dense graphs [Arora et al., JCSS, 1999]:
    - of minimum degree $\Omega(n)$
    - of $\Omega(n^2)$ edges when k is $\Omega(n)$
  - Constant approximation algorithms for complete graphs where the weights satisfy the triangle inequality:
    - 4-approximation [Ravi et al., Oper. Res., 1994]
    - 2-approximation [Hassin et al., Oper. Res. Letters, 1997]

- There is no a PTAS for the DkS problem [Khot, FOCS, 2004]
- There is not known inapproximability result for some approximation ratio
- There are constant approximation algorithms for special cases of the problems:
  - A PTAS for dense graphs [Arora et al., JCSS, 1999]:
    - of minimum degree $\Omega(n)$
    - of $\Omega(n^2)$ edges when k is $\Omega(n)$
  - Constant approximation algorithms for complete graphs where the weights satisfy the triangle inequality:
    - 4-approximation [Ravi et al., Oper. Res., 1994]
    - 2-approximation [Hassin et al., Oper. Res. Letters, 1997]

- DkS is trivial on trees,
- connected HkS on trees can be solved optimally by a dynamic programming algorithm:
  - [Perl and Shiloach, SIAM J. Alg. Discr. Methods, 1983]
  - [Fischetti et al., Networks, 1994]
  - [Goldschmidt and Hochbaum, DAM, 1997]

- DkS is trivial on trees,
- connected HkS on trees can be solved optimally by a dynamic programming algorithm:
  - [Perl and Shiloach, SIAM J. Alg. Discr. Methods, 1983]
  - [Fischetti et al., Networks, 1994]
  - [Goldschmidt and Hochbaum, DAM, 1997]

A generalization of the algorithm(s) yielding a connected solution

- $T = (V, E)$: a weighted tree rooted at some arbitrary vertex s

- T_u: the subtree of T rooted at vertex u, $u \in V - \{s\}$
- $C_u^1(j)$: the value of an optimal solution to HjS on T_u **including** u
- $C_u^0(j)$: the value of an optimal solution to HjS on T_u **excluding** u
- $C_u(j) = \max\{C_u^0(j), C_u^1(j)\}$: the value of an optimal solution to HjS on T_u
- The **optimal solution** is $C_s(k)$

A generalization of the algorithm(s) yielding a connected solution

- $T = (V, E)$: a weighted tree rooted at some arbitrary vertex s
- T_u: the subtree of T rooted at vertex u, $u \in V - \{s\}$
- $C_u^1(j)$: the value of an optimal solution to HjS on T_u **including** u
- $C_u^0(j)$: the value of an optimal solution to HjS on T_u **excluding** u
- $C_u(j) = \max\{C_u^0(j), C_u^1(j)\}$: the value of an optimal solution to HjS on T_u
- The **optimal solution** is $C_s(k)$

A generalization of the algorithm(s) yielding a connected solution

- $T = (V, E)$: a weighted tree rooted at some arbitrary vertex s
- T_u: the subtree of T rooted at vertex u, $u \in V - \{s\}$
- $C_u^1(j)$: the value of an optimal solution to HjS on T_u **including** u
- $C_u^0(j)$: the value of an optimal solution to HjS on T_u **excluding** u
- $C_u(j) = \max\{C_u^0(j), C_u^1(j)\}$: the value of an optimal solution to HjS on T_u
- The **optimal solution** is $C_s(k)$

A generalization of the algorithm(s) yielding a connected solution

- $T = (V, E)$: a weighted tree rooted at some arbitrary vertex s
- T_u: the subtree of T rooted at vertex u, $u \in V - \{s\}$
- $C_u^1(j)$: the value of an optimal solution to HjS on T_u **including** u
- $C_u^0(j)$: the value of an optimal solution to HjS on T_u **excluding** u
- $C_u(j) = \max\{C_u^0(j), C_u^1(j)\}$: the value of an optimal solution to HjS on T_u
- The **optimal solution** is $C_s(k)$

A generalization of the algorithm(s) yielding a connected solution

- $T = (V, E)$: a weighted tree rooted at some arbitrary vertex s
- T_u: the subtree of T rooted at vertex u, $u \in V - \{s\}$
- $C_u^1(j)$: the value of an optimal solution to HjS on T_u **including** u
- $C_u^0(j)$: the value of an optimal solution to HjS on T_u **excluding** u
- $C_u(j) = \max\{C_u^0(j), C_u^1(j)\}$: the value of an optimal solution to HjS on T_u
- The **optimal solution** is $C_s(k)$

A generalization of the algorithm(s) yielding a connected solution

- $T = (V, E)$: a weighted tree rooted at some arbitrary vertex s
- T_u: the subtree of T rooted at vertex u, $u \in V - \{s\}$
- $C_u^1(j)$: the value of an optimal solution to HjS on T_u **including** u
- $C_u^0(j)$: the value of an optimal solution to HjS on T_u **excluding** u
- $C_u(j) = \max\{C_u^0(j), C_u^1(j)\}$: the value of an optimal solution to HjS on T_u
- The **optimal solution** is $C_s(k)$

- The algorithm recursively computes the values $C_u^1(j)$ and $C_u^0(j)$

  -i.e. it solves all HjS problems, $j = 1, 2, ..., k$, for each $u \in V$

- Consider a vertex $u \in V$:
  - $u_1, u_2, ..., u_t$: the children of u in arbitrary order
  - w_i: the weight of the edge (u, u_i), $i = 1, 2, ..., t$
  - $\epsilon \in \{0, 1\}$

- The algorithm recursively computes the values $C_u^1(j)$ and $C_u^0(j)$

  -i.e. it solves all HjS problems, $j = 1, 2, ..., k$, for each $u \in V$

- Consider a vertex $u \in V$:
  - $u_1, u_2, ..., u_t$: the children of u in arbitrary order
  - w_i: the weight of the edge (u, u_i), $i = 1, 2, ..., t$
  - $\epsilon \in \{0, 1\}$

Calculation of $C_u^\epsilon(j)$:

- First, calculate $C_u^\epsilon(j)$ taking into account two children of u

$$C_u^\epsilon(j) = \max_{\substack{k_1+k_2=j-\epsilon, \\ \epsilon_1,\epsilon_2\in\{0,1\}}} \{C_{u_1}^{\epsilon_1}(k_1) + C_{u_2}^{\epsilon_2}(k_2) + \epsilon_1 \cdot \epsilon \cdot w_1 + \epsilon_2 \cdot \epsilon \cdot w_2\}$$

- Then, update $C_u^\epsilon(j)$ for each one of the rest of the children

$$C_u^\epsilon(j) = \max_{\substack{k_1+k_2=j, \\ \epsilon_i\in\{0,1\}}} \{C_u^\epsilon(k_1) + C_{u_i}^{\epsilon_i}(k_2) + \epsilon \cdot \epsilon_i \cdot w_i\}$$

Theorem

There is an $O(nk^2)$ algorithm for the HkS problem on a tree

Corollary (1)

There is an $O(nk)$ algorithm for the HkS problem on a chain

Calculation of $C_u^\epsilon(j)$:

- First, calculate $C_u^\epsilon(j)$ taking into account two children of u

$$C_u^\epsilon(j) = \max_{\substack{k_1+k_2=j-\epsilon, \\ \epsilon_1,\epsilon_2\in\{0,1\}}} \{C_{u_1}^{\epsilon_1}(k_1) + C_{u_2}^{\epsilon_2}(k_2) + \epsilon_1 \cdot \epsilon \cdot w_1 + \epsilon_2 \cdot \epsilon \cdot w_2\}$$

- Then, update $C_u^\epsilon(j)$ for each one of the rest of the children

$$C_u^\epsilon(j) = \max_{\substack{k_1+k_2=j, \\ \epsilon_i\in\{0,1\}}} \{C_u^{\epsilon}(k_1) + C_{u_i}^{\epsilon_i}(k_2) + \epsilon \cdot \epsilon_i \cdot w_i\}$$

Theorem

There is an $O(nk^2)$ algorithm for the HkS problem on a tree

Corollary (1)

There is an $O(nk)$ algorithm for the HkS problem on a chain

Calculation of $C_u^\epsilon(j)$:

- First, calculate $C_u^\epsilon(j)$ taking into account two children of u

$$C_u^\epsilon(j) = \max_{\substack{k_1+k_2=j-\epsilon,\\ \epsilon_1,\epsilon_2\in\{0,1\}}} \{C_{u_1}^{\epsilon_1}(k_1) + C_{u_2}^{\epsilon_2}(k_2) + \epsilon_1 \cdot \epsilon \cdot w_1 + \epsilon_2 \cdot \epsilon \cdot w_2\}$$

- Then, update $C_u^\epsilon(j)$ for each one of the rest of the children

$$C_u^\epsilon(j) = \max_{\substack{k_1+k_2=j,\\ \epsilon_i\in\{0,1\}}} \{C_u^{\epsilon}(k_1) + C_{u_i}^{\epsilon_i}(k_2) + \epsilon \cdot \epsilon_i \cdot w_i\}$$

Theorem

There is an $O(nk^2)$ algorithm for the HkS problem on a tree

Corollary (1)

There is an $O(nk)$ algorithm for the HkS problem on a chain

Corollary (2)

There is an $O(nk)$ algorithm for the HkS problem on a cycle

Let a cycle $C : u_1, u_2, ..., u_n, u_1$, remove an edge of C, say $e = (u_n, u_1)$ and consider the chain $P : u_1, u_2, ..., u_n$

- Solve the HkS on the chain P
  -Let OPT_1 (excludes e) be its optimal solution
  -**If both u_1 and u_n belong to OPT_1:**
  $OPT = OPT_1 + w(e)$

- **Otherwise:**
  - Solve the HkS problem on the chain P **imposing the algorithm to include both u_1 and u_n and the weight $w(e)$**
  - Let OPT_2 be its optimal solution (OPT_2 includes edge e)

- $OPT = \max\{OPT_1, OPT_2\}$

Corollary (2)

There is an $O(nk)$ algorithm for the HkS problem on a cycle

Let a cycle $C : u_1, u_2, ..., u_n, u_1$, remove an edge of C, say $e = (u_n, u_1)$ and consider the chain $P : u_1, u_2, ..., u_n$

- Solve the HkS on the chain P
  -Let OPT_1 (excludes e) be its optimal solution
  -**If both u_1 and u_n belong to OPT_1:**
  $OPT = OPT_1 + w(e)$

- **Otherwise:**
  - Solve the HkS problem on the chain P **imposing the algorithm to include both u_1 and u_n and the weight $w(e)$**
  - Let OPT_2 be its optimal solution (OPT_2 includes edge e)

- $OPT = \max\{OPT_1, OPT_2\}$

Corollary (2)

There is an $O(nk)$ algorithm for the HkS problem on a cycle

Let a cycle $C : u_1, u_2, ..., u_n, u_1$, remove an edge of C, say $e = (u_n, u_1)$ and consider the chain $P : u_1, u_2, ..., u_n$

- Solve the HkS on the chain P
  -Let OPT_1 (excludes e) be its optimal solution
  -**If both u_1 and u_n belong to OPT_1:**
  $OPT = OPT_1 + w(e)$

- **Otherwise:**
  - Solve the HkS problem on the chain P **imposing the algorithm to include both u_1 and u_n and the weight $w(e)$**
  - Let OPT_2 be its optimal solution (OPT_2 includes edge e)

- $OPT = \max\{OPT_1, OPT_2\}$

Corollary (2)

There is an $O(nk)$ algorithm for the HkS problem on a cycle

Let a cycle $C : u_1, u_2, ..., u_n, u_1$, remove an edge of C, say $e = (u_n, u_1)$ and consider the chain $P : u_1, u_2, ..., u_n$

- Solve the HkS on the chain P
  -Let OPT_1 (excludes e) be its optimal solution
  -**If both u_1 and u_n belong to OPT_1:**
  $OPT = OPT_1 + w(e)$

- **Otherwise:**
  - Solve the HkS problem on the chain P **imposing the algorithm to include both u_1 and u_n and the weight $w(e)$**
  - Let OPT_2 be its optimal solution (OPT_2 includes edge e)

- $OPT = \max\{OPT_1, OPT_2\}$

Let $g_i = (V_i, E_i), |V_i| = n_i, 1 \leq i \leq m,$, the connected components of G (g_i is either a cycle or a path)

Our algorithm is in two steps:

1. Solve all HjS problems, $1 \leq j \leq \min\{n_i, k\}$, for each $g_i, 1 \leq i \leq m$, by:
Corollary 1, if g_i is a chain, Corollary 2, if g_i is a cycle

Complexity: For all g_i's, $1 \leq i \leq m$: $O(k \sum_{i=1}^{m} n_i) = O(kn)$.

Let $g_i = (V_i, E_i), |V_i| = n_i, 1 \leq i \leq m,$, the connected components of G (g_i is either a cycle or a path)

Our algorithm is in two steps:

1. Solve all HjS problems, $1 \leq j \leq \min\{n_i, k\}$, for each $g_i, 1 \leq i \leq m$, by:
Corollary 1, if g_i is a chain, Corollary 2, if g_i is a cycle

Complexity: For all g_i's, $1 \leq i \leq m$: $O(k \sum_{i=1}^{m} n_i) = O(kn)$.

Let $g_i = (V_i, E_i), |V_i| = n_i, 1 \leq i \leq m,$, the connected components of G (g_i is either a cycle or a path)

Our algorithm is in two steps:

1. Solve all HjS problems, $1 \leq j \leq \min\{n_i, k\}$, for each $g_i, 1 \leq i \leq m$, by:
Corollary 1, if g_i is a chain, Corollary 2, if g_i is a cycle

Complexity: For all g_i's, $1 \leq i \leq m$: $O(k \sum_{i=1}^{m} n_i) = O(kn)$.

2. Solve the HkS problem on G:

Select a set of optimal solutions to the HjS problems on g_i's such that:

- at most one solution (for some value of j) is selected for each g_i
- the total number of their vertices is k, and
- their total cost is maximized

- C_{ij}, $1 \leq i \leq m$, $1 \leq j \leq k$: the cost of the optimal solution to the HjS problem on g_i
- For g_i's where $k > n_i$: set $C_{ij} = 0$ for $j > n_i$

2. Solve the HkS problem on G:

Select a set of optimal solutions to the HjS problems on g_i's such that:

- at most one solution (for some value of j) is selected for each g_i
- the total number of their vertices is k, and
- their total cost is maximized

- C_{ij}, $1 \leq i \leq m$, $1 \leq j \leq k$: the cost of the optimal solution to the HjS problem on g_i
- For g_i's where $k > n_i$: set $C_{ij} = 0$ for $j > n_i$

2. Solve the HkS problem on G:

Select a set of optimal solutions to the HjS problems on g_i's such that:

- at most one solution (for some value of j) is selected for each g_i
- the total number of their vertices is k, and
- their total cost is maximized

- C_{ij}, $1 \leq i \leq m$, $1 \leq j \leq k$: the cost of the optimal solution to the HjS problem on g_i
- For g_i's where $k > n_i$: set $C_{ij} = 0$ for $j > n_i$

A generalization of the KNAPSACK problem:
-items are partitioned into groups and we have to select at most one item from each group

Using a binary variable y_{ij} our problem is formulated as following:

$$\max \sum_{i=1}^{m} \sum_{j=1}^{k} C_{ij} y_{ij}$$

$$\sum_{j=1}^{k} y_{ij} \leq 1, \quad 1 \leq i \leq m$$

$$\sum_{i=1}^{m} \sum_{j=1}^{k} j y_{ij} \leq k$$

$$y_{ij} \in \{0,1\}, \quad 1 \leq i \leq m, \quad 1 \leq j \leq k$$

A generalization of the KNAPSACK problem:
-items are partitioned into groups and we have to select at most one item from each group

Using a binary variable y_{ij} our problem is formulated as following:

$$\max \sum_{i=1}^{m} \sum_{j=1}^{k} C_{ij} y_{ij}$$

$$\sum_{j=1}^{k} y_{ij} \leq 1, \quad 1 \leq i \leq m$$

$$\sum_{i=1}^{m} \sum_{j=1}^{k} j y_{ij} \leq k$$

$$y_{ij} \in \{0,1\}, \quad 1 \leq i \leq m, \quad 1 \leq j \leq k$$

Complexity:

-this problem can be solved in $O(mk^2)$ time by a dynamic programming algorithm [Laoutaris et al., IPL, 2004]

- $O(nk^2)$ time for the second step

Combining the two steps of our algorithm we obtain

Theorem

There is an $O(nk^2)$ algorithm for the HkS problem on graphs of maximal degree two

Complexity:
-this problem can be solved in $O(mk^2)$ time by a dynamic programming algorithm [Laoutaris et al., IPL, 2004]

- $O(nk^2)$ time for the second step

Combining the two steps of our algorithm we obtain

Theorem

There is an $O(nk^2)$ algorithm for the HkS problem on graphs of maximal degree two

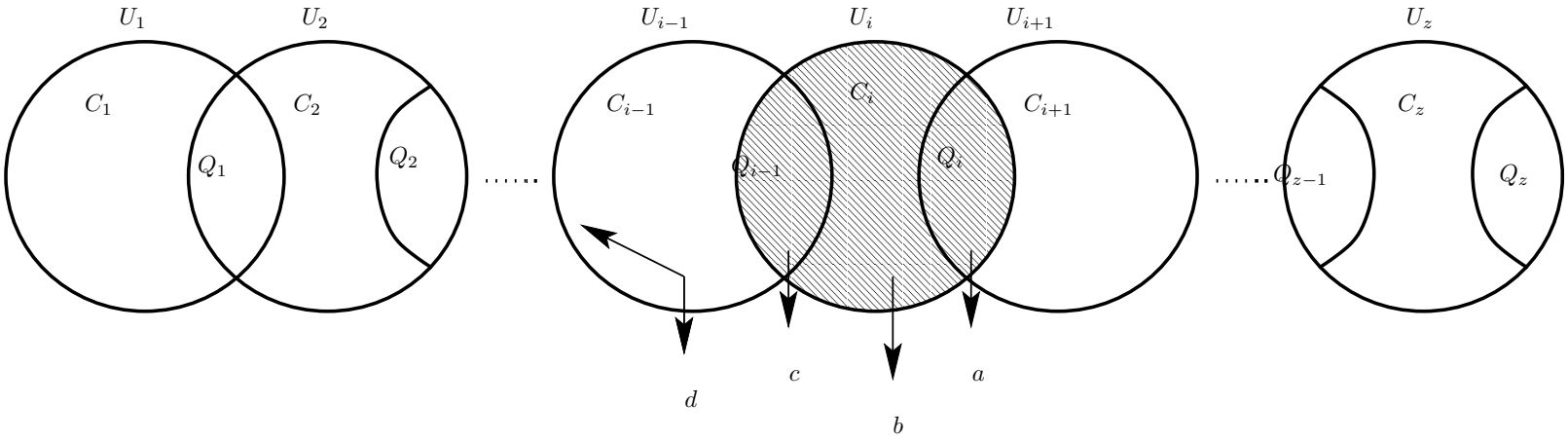
- Interval graph $G = (V, E)$: to each vertex $u \in V$ can be assigned an interval I_u on the real line such that $(u, v) \in E \iff I(u) \bigcap I(v) \neq \emptyset$
- Clique Graph of G: A weighted graph such that:
  - its vertices correspond to maximal cliques, U_i, of G
  - there is an edge (U_i, U_j) iff $U_i \cap U_j \neq \emptyset$
  - $w(U_i, U_j) = |U_i \cap U_j|$
- Clique Tree of G: a maximum weight spanning tree of its clique graph
- All the maximal cliques of an interval graph (and its Clique Graph and Clique Tree) can be found in linear time [Gilmore and Hoffman, Canad. J. Math., 1964]
- The Clique tree of an interval graph is a simple path

- Interval graph $G = (V, E)$: to each vertex $u \in V$ can be assigned an interval I_u on the real line such that $(u, v) \in E \iff I(u) \bigcap I(v) \neq \emptyset$
- Clique Graph of G: A weighted graph such that:
  - its vertices correspond to maximal cliques, U_i, of G
  - there is an edge (U_i, U_j) iff $U_i \cap U_j \neq \emptyset$
  - $w(U_i, U_j) = |U_i \cap U_j|$
- Clique Tree of G: a maximum weight spanning tree of its clique graph
- All the maximal cliques of an interval graph (and its Clique Graph and Clique Tree) can be found in linear time [Gilmore and Hoffman, Canad. J. Math., 1964]
- The Clique tree of an interval graph is a simple path

- Interval graph $G = (V, E)$: to each vertex $u \in V$ can be assigned an interval I_u on the real line such that $(u, v) \in E \iff I(u) \bigcap I(v) \neq \emptyset$
- Clique Graph of G: A weighted graph such that:
  - its vertices correspond to maximal cliques, U_i, of G
  - there is an edge (U_i, U_j) iff $U_i \cap U_j \neq \emptyset$
  - $w(U_i, U_j) = |U_i \cap U_j|$
- Clique Tree of G: a maximum weight spanning tree of its clique graph
- All the maximal cliques of an interval graph (and its Clique Graph and Clique Tree) can be found in linear time [Gilmore and Hoffman, Canad. J. Math., 1964]
- The Clique tree of an interval graph is a simple path

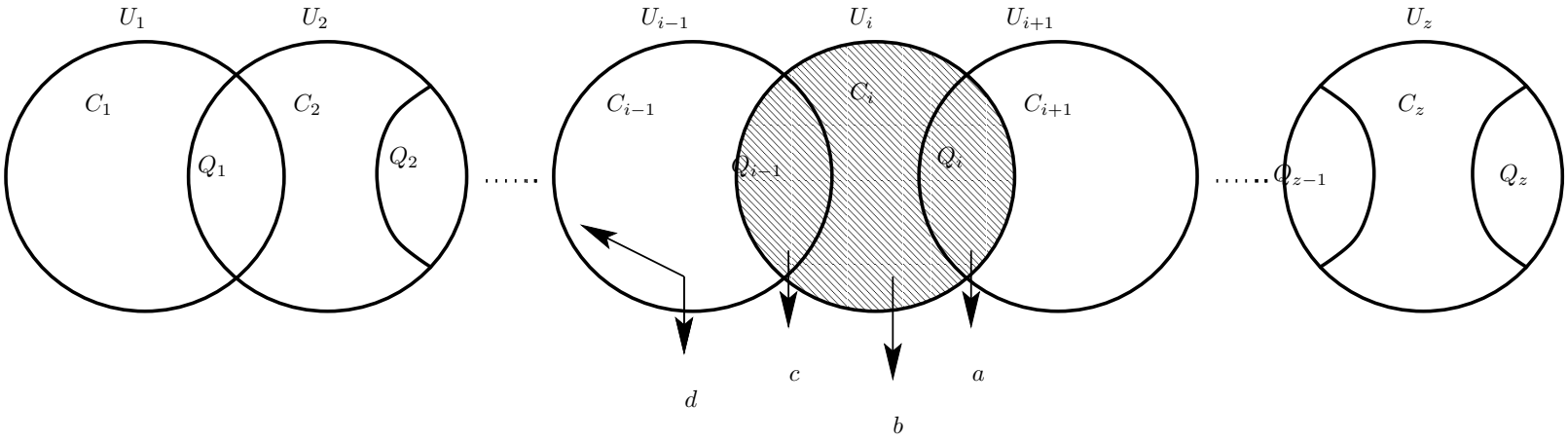
- Interval graph $G = (V, E)$: to each vertex $u \in V$ can be assigned an interval I_u on the real line such that $(u, v) \in E \iff I(u) \bigcap I(v) \neq \emptyset$
- Clique Graph of G: A weighted graph such that:
  - its vertices correspond to maximal cliques, U_i, of G
  - there is an edge (U_i, U_j) iff $U_i \cap U_j \neq \emptyset$
  - $w(U_i, U_j) = |U_i \cap U_j|$
- Clique Tree of G: a maximum weight spanning tree of its clique graph
- All the maximal cliques of an interval graph (and its Clique Graph and Clique Tree) can be found in linear time [Gilmore and Hoffman, Canad. J. Math., 1964]
- The Clique tree of an interval graph is a simple path

- Interval graph $G = (V, E)$: to each vertex $u \in V$ can be assigned an interval I_u on the real line such that $(u, v) \in E \iff I(u) \bigcap I(v) \neq \emptyset$
- Clique Graph of G: A weighted graph such that:
  - its vertices correspond to maximal cliques, U_i, of G
  - there is an edge (U_i, U_j) iff $U_i \cap U_j \neq \emptyset$
  - $w(U_i, U_j) = |U_i \cap U_j|$
- Clique Tree of G: a maximum weight spanning tree of its clique graph
- All the maximal cliques of an interval graph (and its Clique Graph and Clique Tree) can be found in linear time [Gilmore and Hoffman, Canad. J. Math., 1964]
- The Clique tree of an interval graph is a simple path

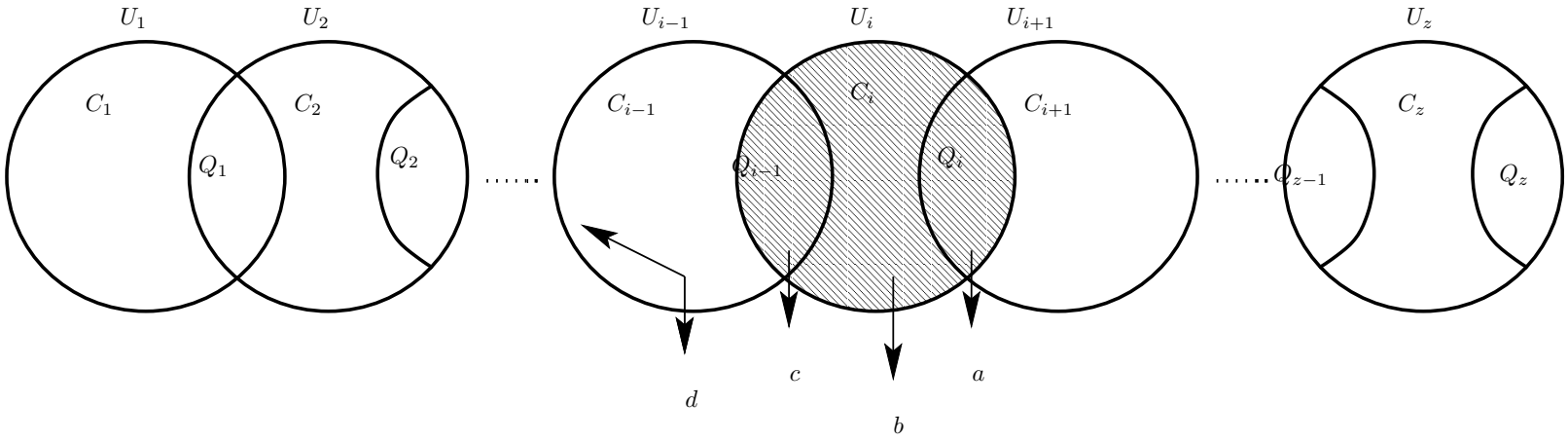


Our subclass: interval graphs whose the clique graph is a simple path

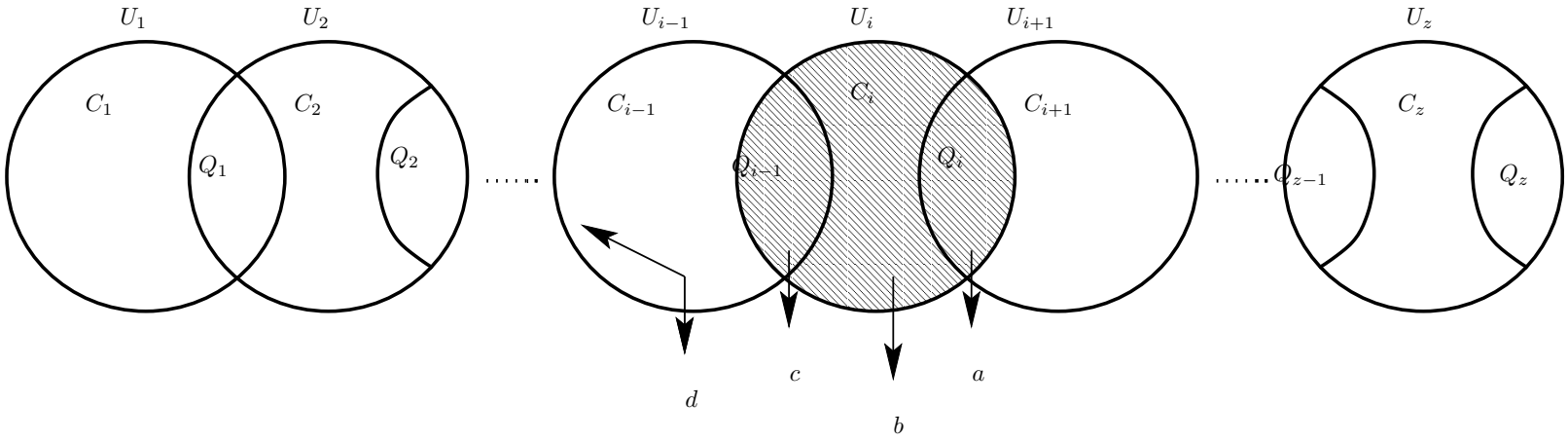
- $U_1, ..., U_z$: the maximal cliques of G
- $Q_i = U_i \cap U_{i+1},\ i = 1, ..., z-1$
- Q_z is a **single vertex** in $U_z - Q_{z-1}$
- $C_1 = U_1 - Q_1,\ C_i = U_i - Q_i - Q_{i-1}, i = 2, ..., z$



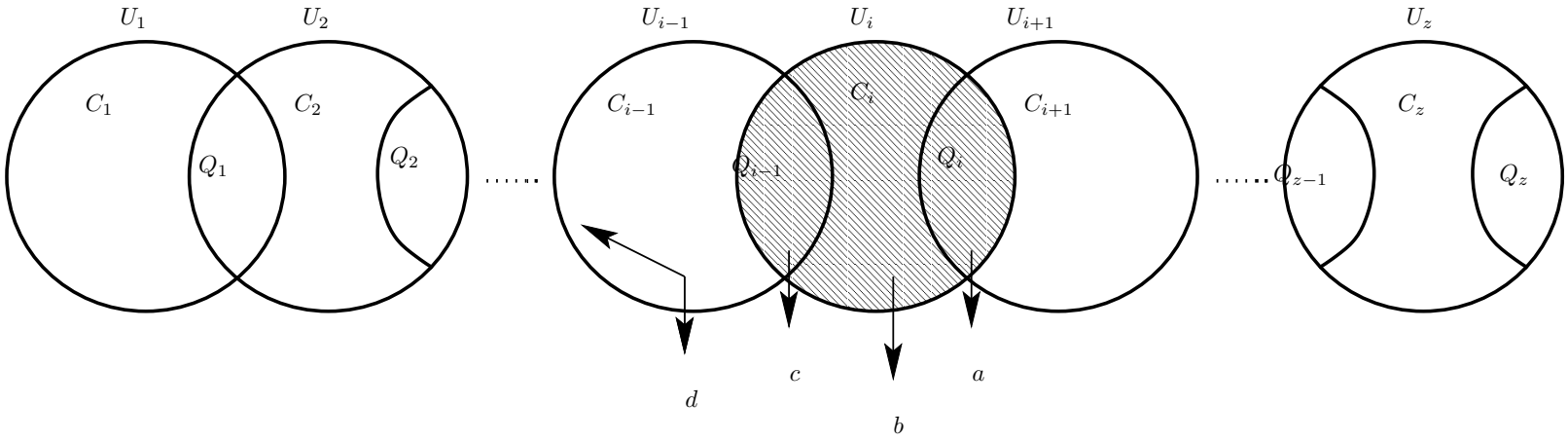
- G_i: the subgraph $U_1, U_2, ..., U_i$ of G
- u_i: a vertex in Q_i, $i = 1, ..., z$ (the "last" vertex of G_i)
- $f_{u_i}(j)$: the value of an optimal solution to DjS on G_i
- it is sufficient to examine only one vertex per Q_i, $i = 1, ..., z$: the value $f_{u_i}(j)$ is the same for all the vertices of Q_i



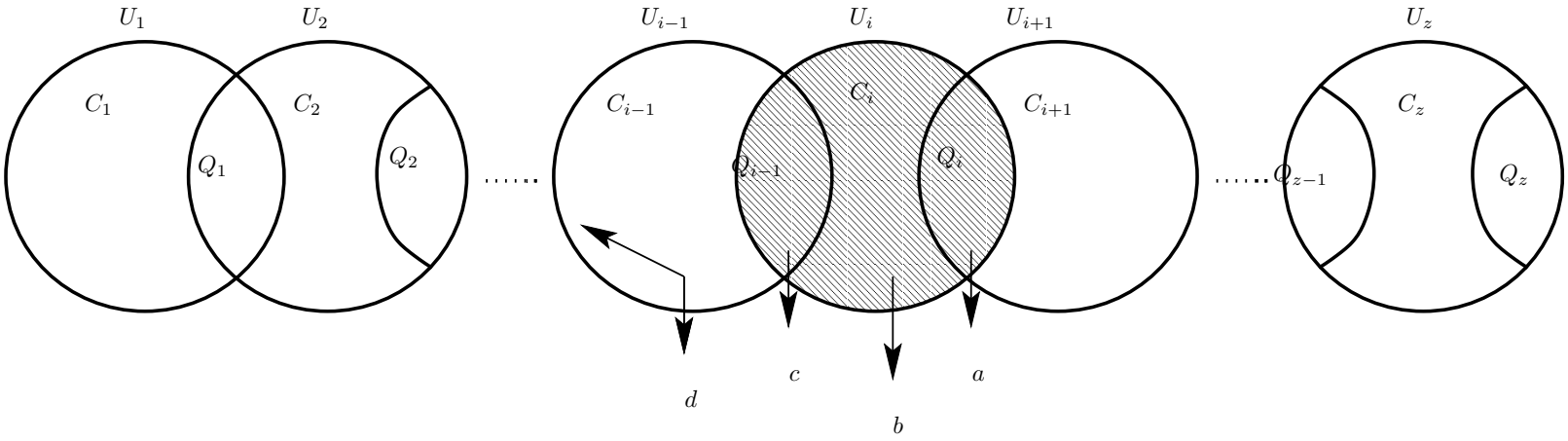
- $f_{u_i}(j, 0)$: the value of an optimal solution to DjS on G_i **not including** vertices of Q_i,
- $f_{u_i}(j, a)$: the value of an optimal solution to DjS on G_i **including exactly** a vertices of Q_i, where $1 \leq a \leq |Q_i|$,
- $f_{u_i}(j) = \max_{0 \leq a \leq |Q_i|} \{f_{u_i}(j, a)\}$



- $f_{u_i}(j, 0)$: the value of an optimal solution to DjS on G_i **not including** vertices of Q_i,
- $f_{u_i}(j, a)$: the value of an optimal solution to DjS on G_i **including exactly** a vertices of Q_i, where $1 \leq a \leq |Q_i|$,
- $f_{u_i}(j) = \max_{0 \leq a \leq |Q_i|} \{f_{u_i}(j, a)\}$



- $f_{u_i}(j, 0)$: the value of an optimal solution to DjS on G_i **not including** vertices of Q_i,
- $f_{u_i}(j, a)$: the value of an optimal solution to DjS on G_i **including exactly** a vertices of Q_i, where $1 \leq a \leq |Q_i|$,
- $f_{u_i}(j) = \max_{0 \leq a \leq |Q_i|} \{f_{u_i}(j, a)\}$



- the algorithm computes recursively the values $f_{u_i}(j)$ for each $i = 1, 2, ..., z$ and each $j = 1, 2, ..., k$
- the optimal solution is $f_{u_z}(k) = \max_{a=0,1}\{f_{u_z}(k, a)\}$, since $|Q_z| = 1$.

For $u_i \in Q_i, 2 \leq i \leq z$, $1 \leq j \leq k$

- For $a = 0$:

$$f_{u_i}(j, a) = f_{u_{i-1}}(j)$$

- For $1 \leq a \leq |Q_i|$:

$$f_{u_i}(j, a) = \begin{cases} j(j-1)/2, & \text{if } \; j \leq |U_i - Q_i| + a \\ \max\limits_{a+b+c+d=j} \{f_{u_{i-1}}(c+d, c) + \binom{a+b}{2} + c(a+b)\}, & \text{otherwise} \end{cases}$$

where $0 \leq b \leq |C_i|$, $1 \leq c \leq |Q_{i-1}|$, $0 \leq d \leq |\bigcup_{i=1}^{i-1} U_i - Q_{i-1}|$

For $u_i \in Q_i, 2 \leq i \leq z$, $1 \leq j \leq k$

- For $a = 0$:

$$f_{u_i}(j, a) = f_{u_{i-1}}(j)$$

- For $1 \leq a \leq |Q_i|$:

$$f_{u_i}(j,a) = \begin{cases} j(j-1)/2, & \text{if } \; j \leq |U_i - Q_i| + a \\ \max\limits_{a+b+c+d=j} \{f_{u_{i-1}}(c+d, c) + \binom{a+b}{2} + c(a+b)\}, & \text{otherwise} \end{cases}$$

where $0 \leq b \leq |C_i|$, $1 \leq c \leq |Q_{i-1}|$, $0 \leq d \leq |\bigcup_{i=1}^{i-1} U_i - Q_{i-1}|$

Theorem

There is an $O(nk^4)$ algorithm for the connected DkS problem on interval graphs whose the clique graph is a simple path

Time to compute all $f_{u_i}(j)$:

- for each combination of a, b, c, d such that $a + b + c + d = j \leq k$: $O(k^3)$
- for all j's, $j = 1, 2, ..., k$: $O(k^4)$
- for all u_i's, $i = 1, 2, ..., z \leq n$: $O(nk^4)$

- What is the complexity of the DkS problem on:
  - Interval graphs
  - Proper interval graphs
  - Permutation graphs
- Approximation algorithms for special graph classes:
  - Bipartite graphs
  - Comparability graphs
  - Chordal graphs
  - Planar graphs
  - Bounded degree graphs, (even bipartite)
  - Regular graphs
- Better approximation algorithms for arbitrary graphs