Graph pattern matching: обзор

Костюков Юрий, 371

Graph pattern matching: необходимость изучения

- Распознавание паттернов
- Классификация сайтов, социальных групп,…
- Нахождение сообществ
- Распознавание социальной иерархии
- Социальный маркетинг
- …

Фундаментальные понятия и алгоритмы

Здесь и далее изображения местами взяты из лекций Wenfei Fan: QSX
http://homepages.inf.ed.ac.uk/wenfei/qsx/home.html

Graph pattern matching: определение

Параметры: ориентированный граф G и граф-паттерн Q

Результат: все подграфы G, “удовлетворяющие” Q

Методы graph pattern matching’а различаются определением понятия “удовлетворять”

На вершине башни: изоморфизм подграфов

Напоминание: это такая биекция f из вершин Q в вершины G, что

- в Q есть ребро (u, u’) ⇔ в G есть ребро (f(u), f(u’))
- на u из Q и f(u) метки совпадают

Параметры: ориентированный граф G и граф-паттерн Q

Результат: все подграфы G изоморфные Q

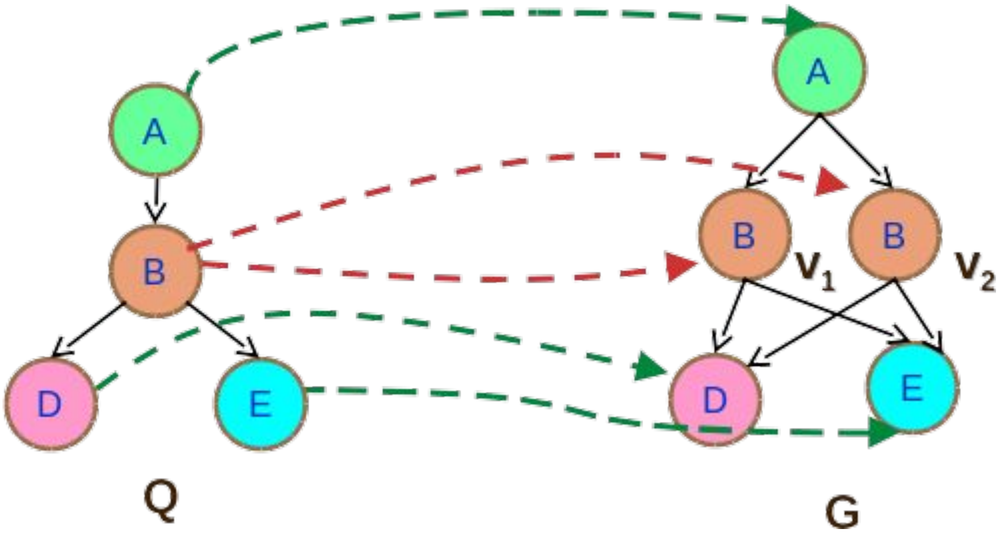
1. NP сложная
    a. даже для таких частных случаев, как:
    b. Q — лес, G — дерево
    c. Q — дерево, G — ацикличный граф
2. PTIME если Q — дерево и G — лес

Согласно Garey, M.R. and Johnson, D.S., 2002. Computers and intractability (Vol. 29).

Первое приближение: graph simulation

Отношение S на вершинах Q и G, такое что:

- $\forall u \in Q.V$: $\exists v \in G.V$: $(u, v) \in S$ & метки u и v совпадают
- $\forall (u, v) \in S$: $\forall (u, u') \in Q.E$: $\exists (v, v') \in G.E$: $(u', v') \in S$



Graph simulation: свойства

Параметры: ориентированный граф G и граф-паттерн Q

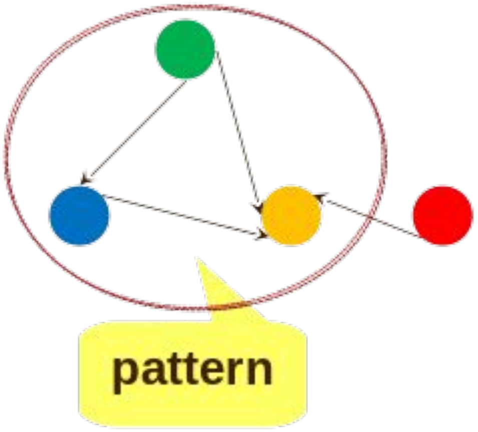
Результат: максимальное отношение симуляции S

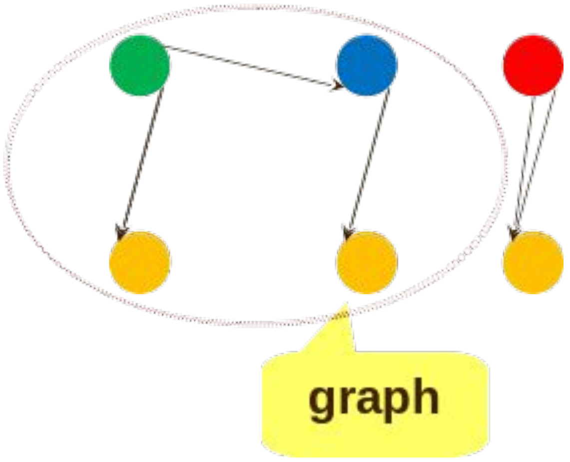
1. R всегда существует и единственно
2. Сложность: $O((|G.V| + |Q.V|)(|G.E| + |Q.E|))$
3. **Теорема:** Если в Q есть *ориентированный* цикл, то и в матче тоже

Graph simulation: ограничения

$\forall (u, v) \in S: \forall (u, u') \in Q.E: \exists (v, v') \in G.E: (u', v') \in S$ — *сохранение детей*

- связный паттерн матчит несвязный граф
- цикл матчит дерево
- родители не сохраняются

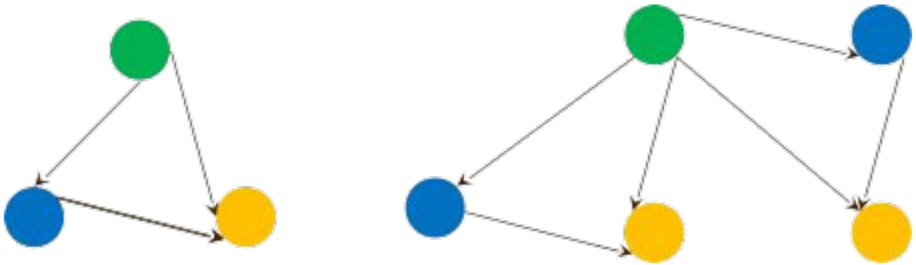




Второе приближение: dual simulation

Отношение S на вершинах Q и G, такое что $\forall (u, v) \in S$:

- $\forall (u, u') \in Q.E$: $\exists (v, v') \in G.E$: $(u', v') \in S$ — *дети*
- $\forall (u', u) \in Q.E$: $\exists (v', v) \in G.E$: $(u', v') \in S$ — *родители*



Dual simulation: свойства

1. **Лемма:** Если G дуально симулирует Q, то G симулирует Q (очевидно)
2. **Теорема:** Если в Q есть *неориентированный* цикл, то и в матче тоже
3. **Теорема:** Если G дуально симулирует Q, то каждая компонента связности Q матчит ровно одну компоненту связности G

Последнее приближение: strong simulation

Отношение S на вершинах Q и G, такое что $\exists v \in G.V$, $Gs \subseteq G$:

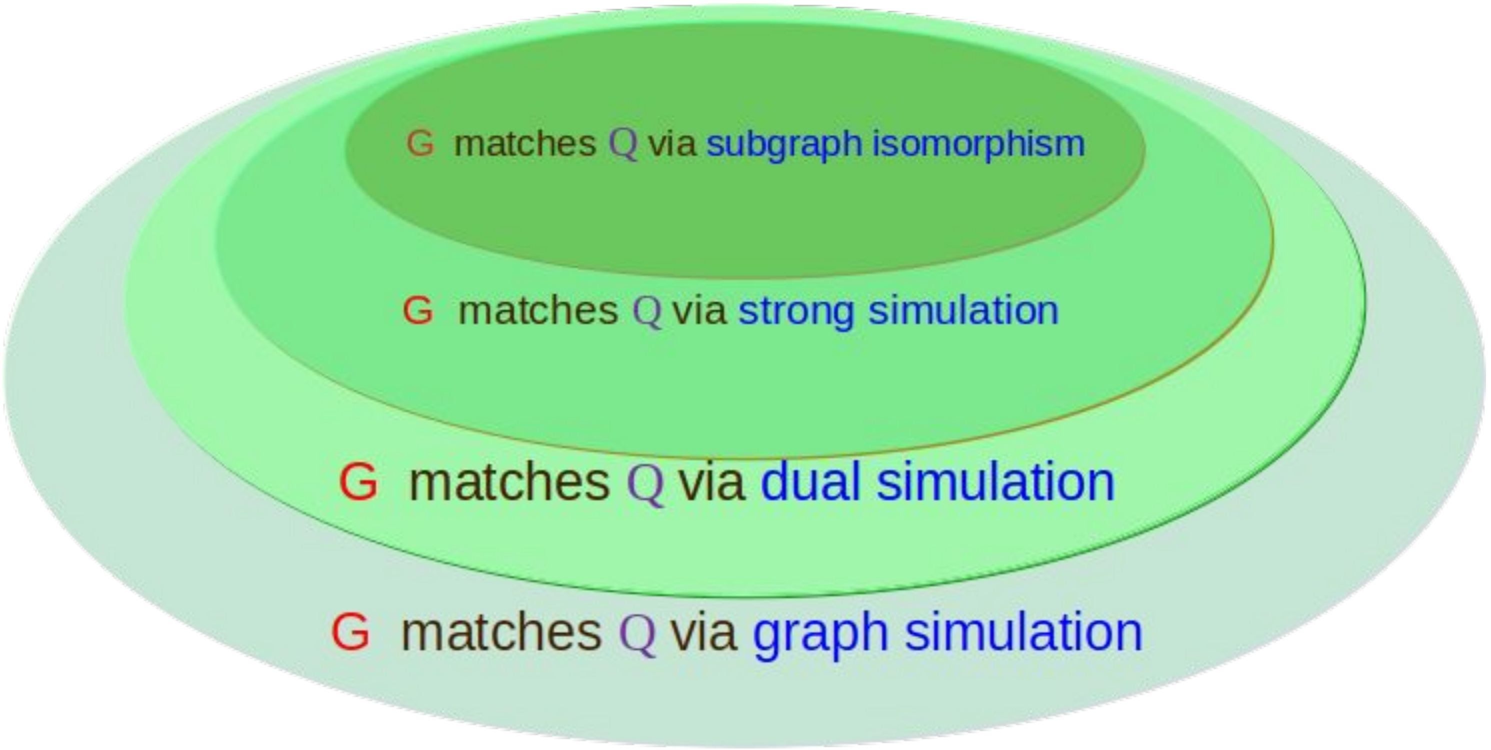
- S — dual simulation
- Gs — match подграф G с т.з. S
- *Locality:* Gs лежит в шаре G[v, dQ]: $v \in Gs$, dQ — диаметр Q

Шар G[v, r] — вершины с соотв. рёбрами, стоящие от v на расстоянии не более r.

Strong simulation: свойства

1. **Лемма:** Если G сильно симулирует Q, то G дуально симулирует Q (очевидно)
2. **Теорема:** Если G сильно симулирует Q, то диаметр матча не превосходит удвоенного диаметра Q

Иерархия подходов



Сложность

Подход	Сложность
изоморфизм подграфов	NP-полная
graph simulation	квадратичное время (от |V|)
dual simulation	кубическое время (от |V|)
strong simulation	кубическое время (от |V|)

Итоговые свойства

Table II. Topology preservation and bounded matches

Property	Matching			
	Graph simulation	Dual simulation	Strong simulation	Subgraph isomorphism
Children	✓	✓	✓	✓
Parents	×	✓	✓	✓
Weak Connectivity	×	✓	✓	✓
Strong Connectivity	×	×	✓	✓
Directed cycles	✓	✓	✓	✓
Undirected cycles	×	✓	✓	✓
Locality	×	×	✓	✓
Bounded Matches	✓	×	✓	×
Bisimilarity	×	×	×	✓
Bounded cycles	×	×	×	✓

Таблица из статьи:
Ma, S., Cao, Y., Fan, W., Huai, J. and Wo, T., 2014. Strong simulation: Capturing topology in graph pattern matching.

Итоговые свойства

Table II. Topology preservation and bounded matches

Property	Matching			
	Graph simulation	Dual simulation	Strong simulation	Subgraph isomorphism
Children	✓	✓	✓	✓
Parents	×	✓	✓	✓
Weak Connectivity	×	✓	✓	✓
Strong Connectivity	×	×	✓	✓
Directed cycles	✓	✓	✓	✓
Undirected cycles	×	✓	✓	✓
Locality	×	×	✓	✓
Bounded Matches	✓	×	✓	×
Bisimilarity	×	×	×	✓
Bounded cycles	×	×	×	✓

coNP-hard!

Оптимизации для больших графов

Распределённый pattern matching: проблемы

- графы большие (от десятков миллионов до миллиардов вершин)
- графы постоянно меняются
    - удаляются / добавляются пользователи
    - добавить в друзья / кинуть в ЧС
    - …

Распределённый pattern matching: подходы

- Параллельное исполнение запросов
- Ограниченно вычисляемые запросы (Bounded evaluable queries)
- Сжатие графа (Query-preserving graph compression)
- Выражение запроса через базисные (Query answering using views)
- Ограниченный инкрементальный graph pattern matching (Bounded incremental graph pattern matching)

Bounded evaluable queries

Параметры: набор запросов Q, схема доступа A
Схема доступа: “у каждого фильма не более 30 актёров”
Результат:

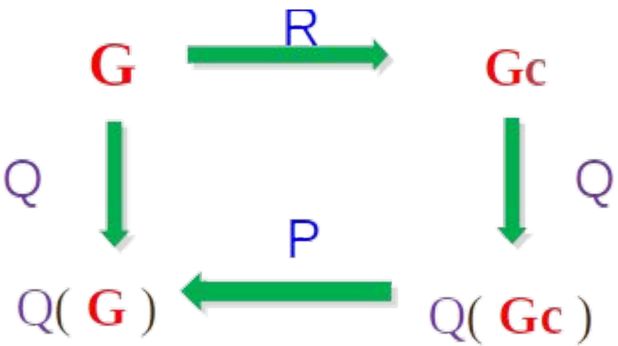
- |Gq| — независим от |G|
- Q(G) = Q(Gq)
- Gq находится за время, зависящее только от Q и A
    - $O(\ |A|\ |Vq|\ |Eq| + |A|\ |Eq| + |A|\ |Vq|^2\ )$

Работает не для всякого Q, но на практике около 60% запросов удовлетворяют.

⇒ авторы получили ускорение **в 28587 раз**

Query-preserving graph compression

1. По классу запросов L создаём функции сжатия и расжатия R и P.
2. Offline: Gc := R(G)
   a. | Gc | << | G |
3. Online:
   a. Считаем Q(Gc) — любым алгоритмом
   b. Q(G) == P(Q(Gc))



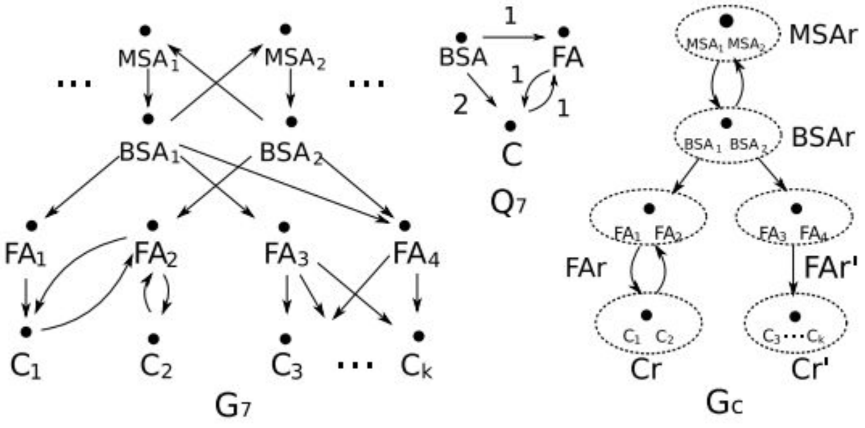


Figure 7: A social graph and its compression

Примеры классов запросов

- *Достижимость*
  - **Задача:** существует ли путь из s в t в G?
  - **R:** отображаем вершины в компоненты сильной связности
  - **Сжатие:** 95% в среднем
- *Симуляция*
  - **Задача:** максимальное отношение симуляции R
  - **R:** находим классы эквивалентности — O(|E| log |V|)
  - **P:** O(|Q(G)|)
  - **Сжатие:** 57% в среднем

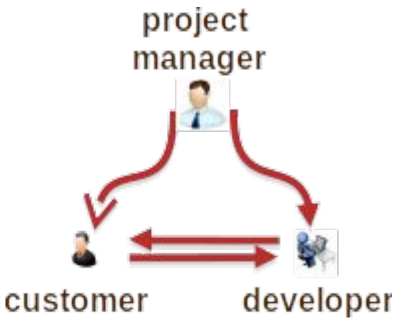
Query answering using views

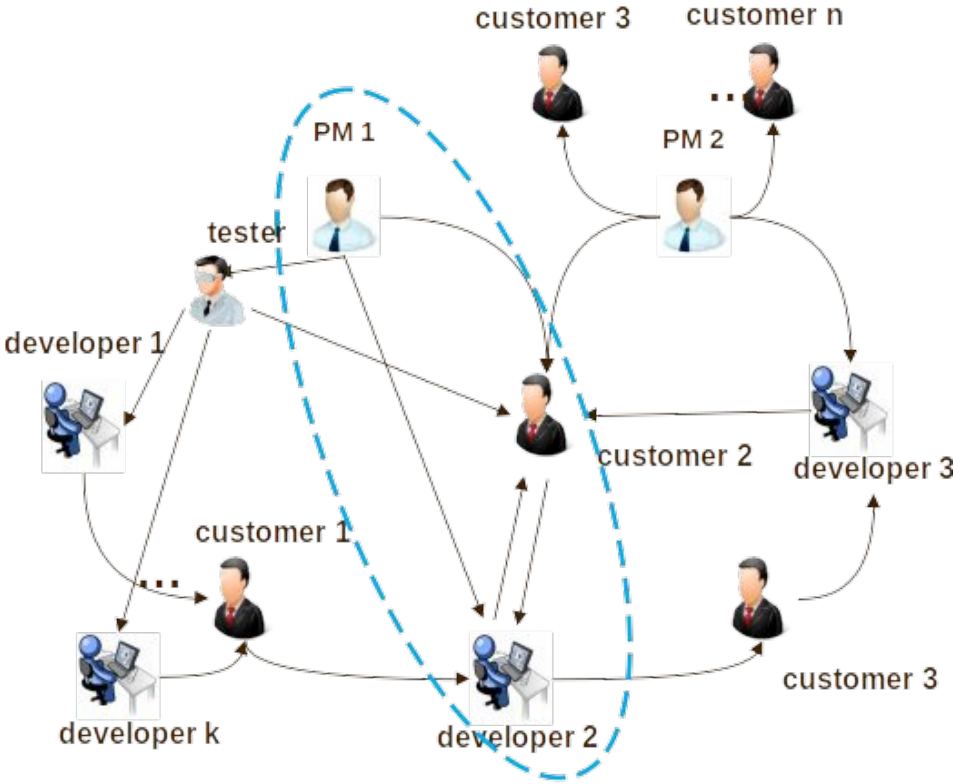
Q содержится в V = {V1, … , Vn} если существует отображение ƛ из рёбер запроса в рёбра V: для любого графа G: рёбра, которые матчатся Q в G, должны матчиться их образами ƛ.

Алгоритм проверки:

1. Сматчить Vi в Q.
2. Проверить, что объединение всех сматченных Vi содержит все рёбра Q

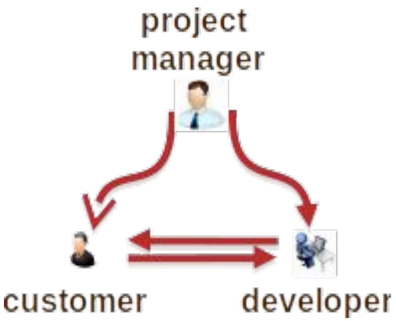
Query answering using views

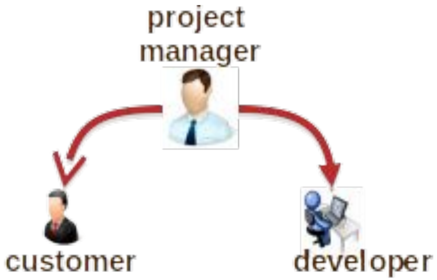




A collaborative (chat) network

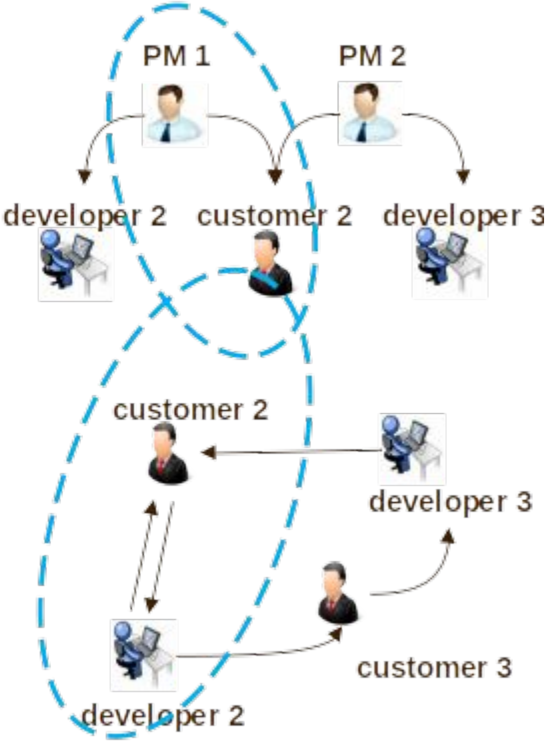
Query answering using views





query 1

query 2



Query answering using views

Примерный алгоритм:

1. Находим оптимальное покрытие V для входного запроса Q ⇒ V
2. Каждый паттерн в покрытии матчим в G ⇒ V(G)
3. Комбинируем результаты V(G) ⇒ Q(G)

Query answering using views

Проверка: $O(\ \mathrm{card}(V)|Q|^2 + |V|^2 + |Q||V|\ )$
Матчинг: $O(\ |Q||V(G)| + |V(G)|^2\ )$

Эффективность зависит от выборки V — её можно выбирать достаточно хорошо с той же ассимптотикой

⇒ авторы получили ускорение **в 23 раза**

Bounded incremental graph pattern matching

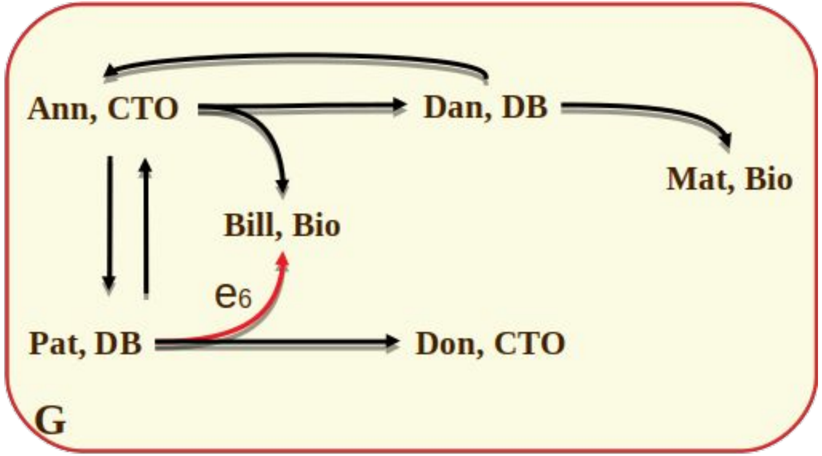
- Реальные данные постоянно меняются (соцсети, БД, …)
- Пересчитывать $Q(G \oplus \Delta G)$ с нуля странно
- ΔG обычно невелико

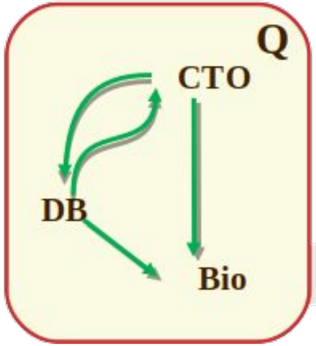
⇒ **Инкрементальный алгоритм:**

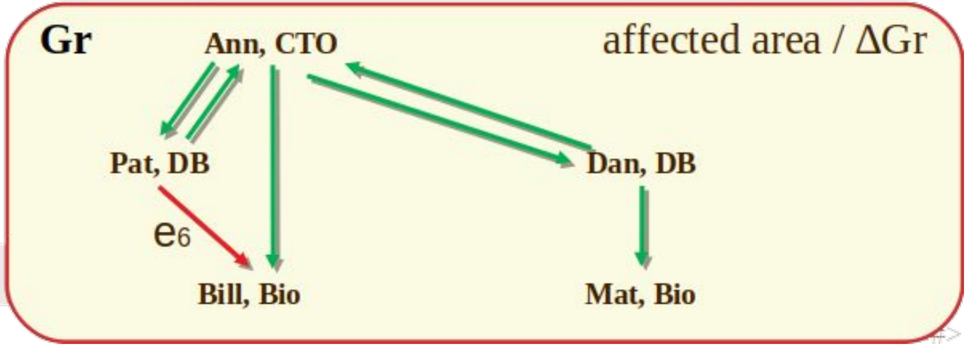
- по Q, G, $Q(G)$, ΔG
- строим ΔM, такой что: $Q(G \oplus \Delta G) = Q(G) \oplus \Delta M$

Bounded incremental graph pattern matching

1. Удаляем e6

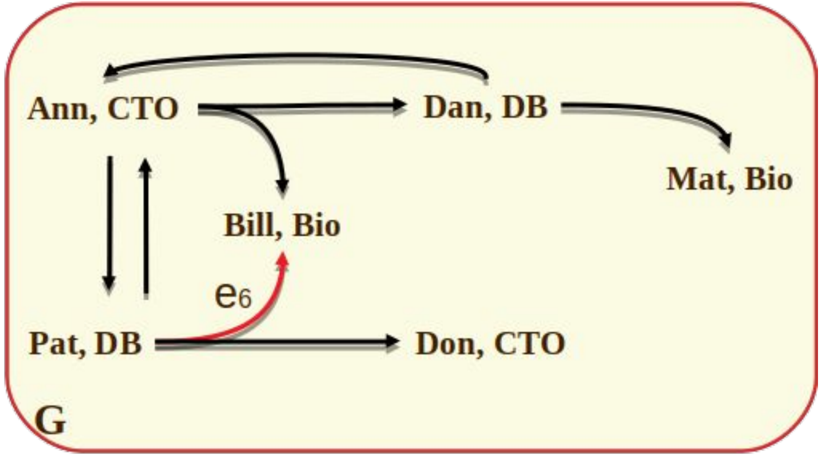


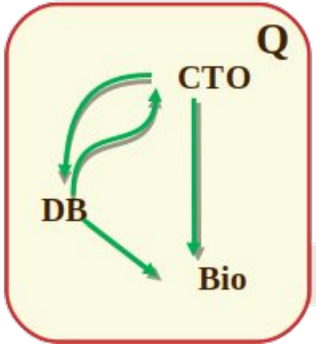


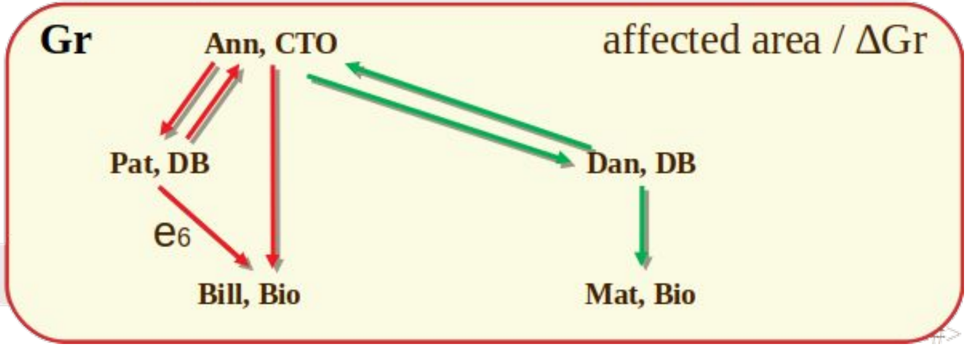


Bounded incremental graph pattern matching

1. Удаляем e6
2. Находим затронутый матч

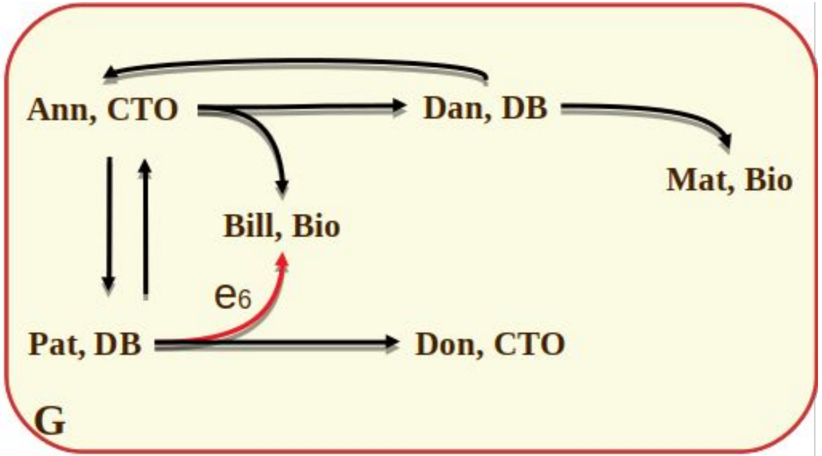


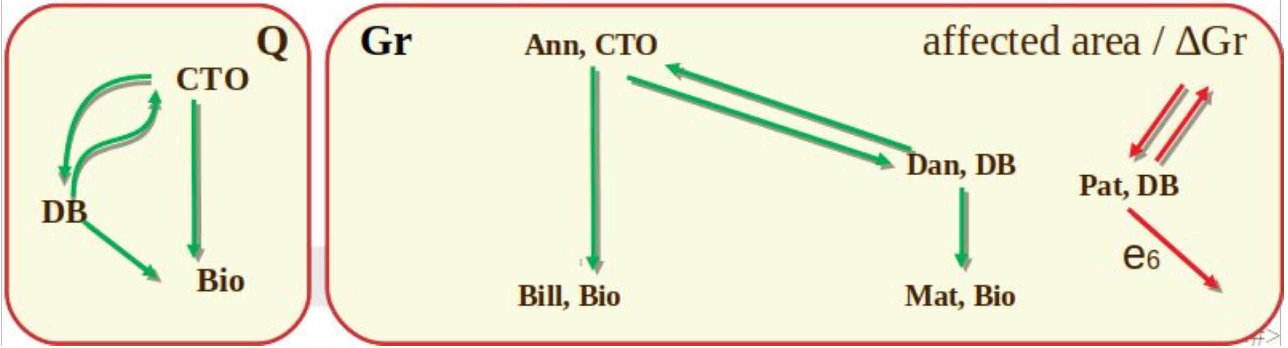




Bounded incremental graph pattern matching

1. Удаляем e6
2. Находим затронутый матч
3. Очищаем





Результаты добавления инкрементальности

- Достижимость (со сжатием): $O(|AFF||Gc|)$
- Сжатие вообще: $O(|AFF|^2 + |Gc|)$
- Симуляция: $O(|\Delta G| (|Q||AFF| + |AFF|^2))$

$\Rightarrow$ ускорение в среднем в 2 раза

AFF — “affected area”

Результаты

- в 28587 раз (bounded evaluability)
- в 23 раза быстрее (query answering using views)
- в 2.3 раза быстрее (сжатие)
- в 2 раза быстрее (инкрементальность)

⇒ сократили 5.28 года * 365 * 24 * 3600 до 24 секунд

Источники

- Garey, M.R. and Johnson, D.S., 2002. Computers and intractability (Vol. 29).
- M. R. Henzinger, T. Henzinger, and P. Kopke, 1995. Computing simulations on finite and infinite graphs. (graph simulation)
- Ma, S., Cao, Y., Fan, W., Huai, J. and Wo, T., 2014. Strong simulation: Capturing topology in graph pattern matching. (strong simulation)
- Ramalingam, G. and Reps, T., 1996. On the computational complexity of dynamic graph problems. (incremental graph algorithms)
- Cao, Y., Fan, W., Huai, J. and Huang, R., 2015, April. Making pattern queries bounded in big graphs.
- Fan, W., 2012, March. Graph pattern matching revised for social network analysis.
- Fan, W., Wang, X. and Wu, Y., 2013. Incremental graph pattern matching.

Appendix: cubic algorithm for strong simulation

Algorithm Match(Q, G)

Input: Pattern graph Q with diameter d_Q and data graph $G(V, E)$.
Output: The set Θ of maximum perfect subgraphs of G for Q.

1. $\Theta := \emptyset$;
2. **for each** ball $\hat{G}[w, d_Q]$ in G **do**
3. $\quad S_w :=$ DualSim$(Q, \hat{G}[w, d_Q])$;
4. $\quad G_s :=$ ExtractMaxPG$(Q, \hat{G}[w, d_Q], S_w)$;
5. $\quad$ **if** $G_s \neq nil$ **then**
6. $\quad\quad \Theta := \Theta \cup \{G_s\}$;
7. **return** Θ.

Procedure ExtractMaxPG$(Q, \hat{G}[w, d_Q], S_w)$

Input: Pattern Q, ball $\hat{G}[w, d_Q]$ with maximum match relation S_w.
Output: The maximum perfect subgraph G_s in $\hat{G}[w, d_Q]$ for Q if any.

1. **if** w does not appear in S_w **then**
2. $\quad$ **return** *nil*;
3. Construct the matching graph G_m *w.r.t.* S_w;
4. **return** the connected component G_s containing w in G_m.

Procedure DualSim$(Q, \hat{G}[w, d_Q])$

Input: Pattern graph $Q(V_q, E_q)$ and ball $\hat{G}[w, d_Q]$.
Output: The maximum match relation S_w in $\hat{G}[w, d_Q]$ for Q.

1. **for each** $u \in V_q$ in Q **do**
2. $\quad \mathsf{sim}(u) := \{v \mid v$ is in $\hat{G}[w, d_Q]$ **and** $l_Q(u) = l_G(v)\}$;
3. **while** there are changes **do**
4. $\quad$ **for each** edge (u, u') in E_Q and **each** node $v \in \mathsf{sim}(u)$ **do**
5. $\quad\quad$ **if** there is no edge (v, v') in $\hat{G}[w, d_Q]$ with $v' \in \mathsf{sim}(u')$ **then**
6. $\quad\quad\quad \mathsf{sim}(u) := \mathsf{sim}(u) \setminus \{v\}$;
7. $\quad$ **for each** edge (u', u) in E_Q and **each** node $v \in \mathsf{sim}(u)$ **do**
8. $\quad\quad$ **if** there is no edge (v', v) in $\hat{G}[w, d_Q]$ with $v' \in \mathsf{sim}(u')$ **then**
9. $\quad\quad\quad \mathsf{sim}(u) := \mathsf{sim}(u) \setminus \{v\}$;
10. $\quad$ **if** $\mathsf{sim}(u) = \emptyset$ **then return** $\emptyset$;
11. $S_w := \{(u, v) \mid u \in V_q, v \in \mathsf{sim}(u)\}$;
12. **return** S_w.