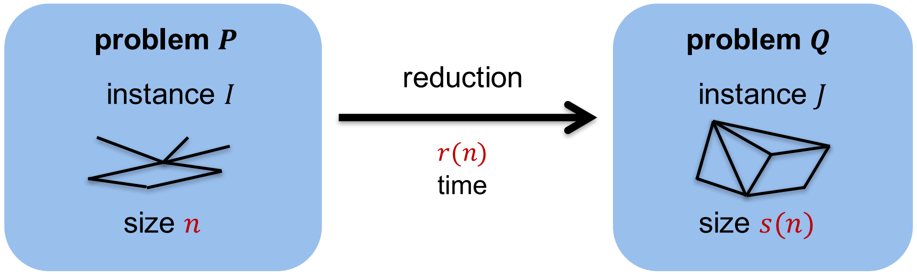
Reminder: Relations = Reductions

transfer hardness of one problem to another one by reductions

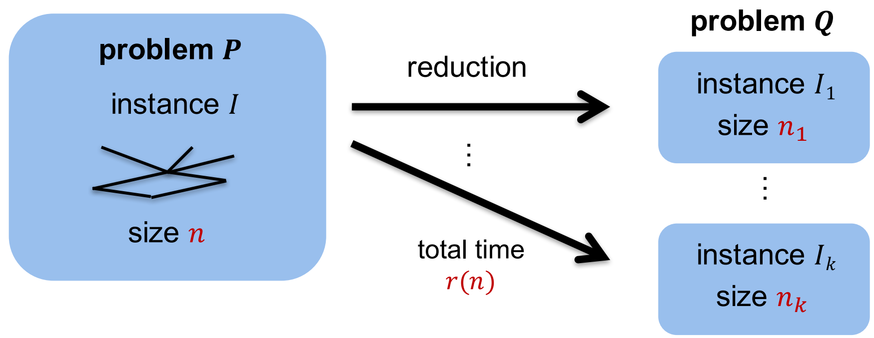


I is a 'yes'-instance $\iff$ J is a 'yes'-instance

$t(n)$ algorithm for Q implies a $r(n) + t(s(n))$ algorithm for P

if P has no $r(n) + t(s(n))$ algorithm then Q has no $t(n)$ algorithm

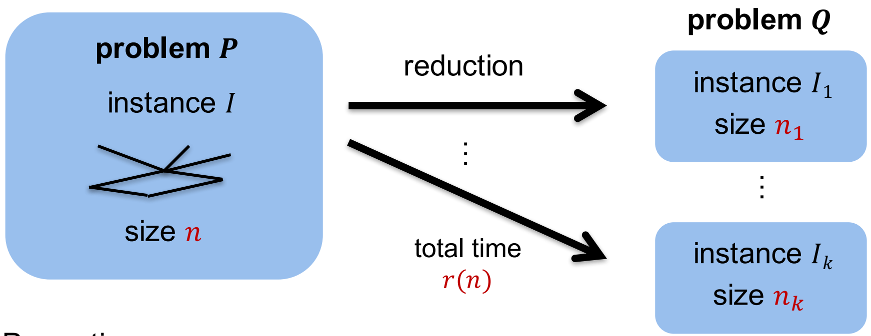
Reminder: Relations = Reductions



Subcubic Reduction

A **subcubic reduction** from P to Q is

an algorithm A for P with **oracle** access to Q s.t.:



Properties:

for any instance I, algorithm $A(I)$ correctly solves problem P on I

A runs in time $r(n) = O(n^{3-\gamma})$ for some $\gamma > 0$

for any $\varepsilon > 0$ there is a $\delta > 0$ s.t. $\sum_{i=1}^{k} n_i^{3-\varepsilon} \leq n^{3-\delta}$

Subcubic Reduction

A **subcubic reduction** from P to Q is

an algorithm A for P with **oracle** access to Q with:

A subcubic reduction implies:

If Q has an $O(n^{3-\alpha})$ algorithm for some $\alpha > 0$,

then P has an $O(n^{3-\beta})$ algorithm for some $\beta > 0$

Properties:

for any instance I, algorithm $A(I)$ correctly solves problem P on I

A runs in time $r(n) = O(n^{3-\gamma})$ for some $\gamma > 0$

for any $\varepsilon > 0$ there is a $\delta > 0$ s.t. $\sum_{i=1}^{k} n_i^{3-\varepsilon} \leq n^{3-\delta}$

similar: subquadratic/subquartic reductions

Subcubic Reduction

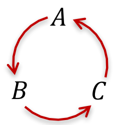
subcubic reduction: write $P \leq Q$

subcubic equivalent: write $P \equiv Q$ if $P \leq Q$ and $Q \leq P$

Transitivity: (Exercise)

For problems A, B, C with $A \leq B$ and $B \leq C$ we have $A \leq C$.

In particular: If $A \leq B$ and $B \leq C$ and $C \leq A$ then A, B, C are subcubic equivalent.



Lemma: (without proof)

If $A \leq B$ and B is in time $O\left(n^3/2^{\Omega(\log n)^{1/2}}\right)$

then A is in time $O\left(n^3/2^{\Omega(\log n)^{1/2}}\right)$.

max planck institut informatik

Reminder

All-Pairs-Shortest-Paths (APSP):

given a weighted directed graph G, compute the (length of the) **shortest path between any pair** of vertices

each edge has a weight in $\{1,..,n^c\}$

Floyd-Warshall'62: $O(n^3)$

...

Williams'14: $O\left(n^3/2^{\Omega(\log n)^{1/2}}\right)$

Conjecture: for any $\varepsilon > 0$ APSP has no $O(n^{3-\varepsilon})$ algorithm

there exists $c > 0$ such that

max planck institut informatik

Reminder

Min-Plus Matrix Product:

each entry in $\{1,..,n^c,\infty\}$

given $n_1 \times n_2$-matrix A and $n_2 \times n_3$-matrix B, define their min-plus product as the $n_1 \times n_3$-matrix C with

$$C_{i,j} = \min_{1 \leq k \leq n_2} A_{i,k} + B_{k,j}$$

from definition: $O(n^3)$ (if $n = n_1 = n_2 = n_3$)

Conjecture: for any $\varepsilon > 0$ there is no $O(n^{3-\varepsilon})$ algorithm

there exists $c > 0$ such that

max planck institut informatik

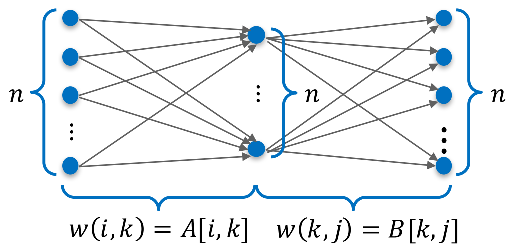
Reminder

Thm:

If APSP has a $T(n)$ algorithm then Min-Plus Product has an $O(T(n) + n^2)$ algorithm.

Thm:

If Min-Plus Product has a $T(n)$ algorithm then APSP has an $O((T(n) + n^2)\log n)$ algorithm.



Consider adjacency matrix A of G

Add selfloops with cost 0: $A + I$

Square $\lceil \log n \rceil$ times using Min-Plus Product: $B := (A + I)^{2^{\lceil \log n \rceil}}$

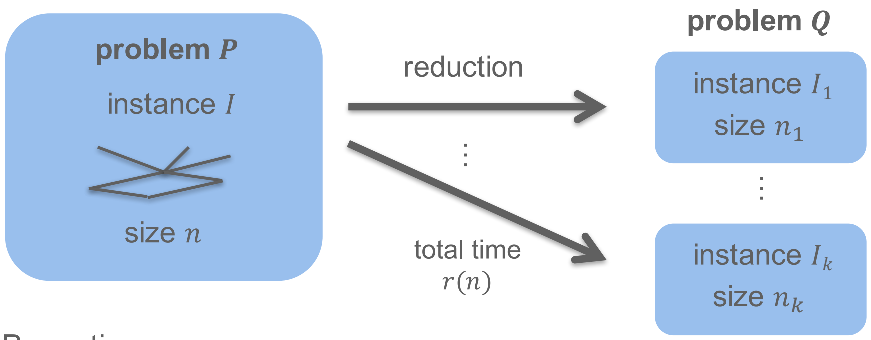
Then $B_{i,j}$ is the length of the shortest path from i to j

max planck institut informatik

Subcubic Reduction

A **subcubic reduction** from P to Q is

an algorithm A for P with **oracle** access to Q with:



Properties:

for any instance I, algorithm $A(I)$ correctly solves problem P on I

A runs in time $r(n) = O(n^{3-\gamma})$ for some $\gamma > 0$

for any $\varepsilon > 0$ there is a $\delta > 0$ s.t. $\sum_{i=1}^{k} {n_i}^{3-\varepsilon} \leq n^{3-\delta}$

max planck institut informatik

Subcubic Equivalences

Thm: If APSP has a $T(n)$ algorithm then Min-Plus Product has an $O(T(n))$ algorithm.

Thm: If Min-Plus Product has a $T(n)$ algorithm then APSP has an $O(T(n) \log n)$ algorithm.

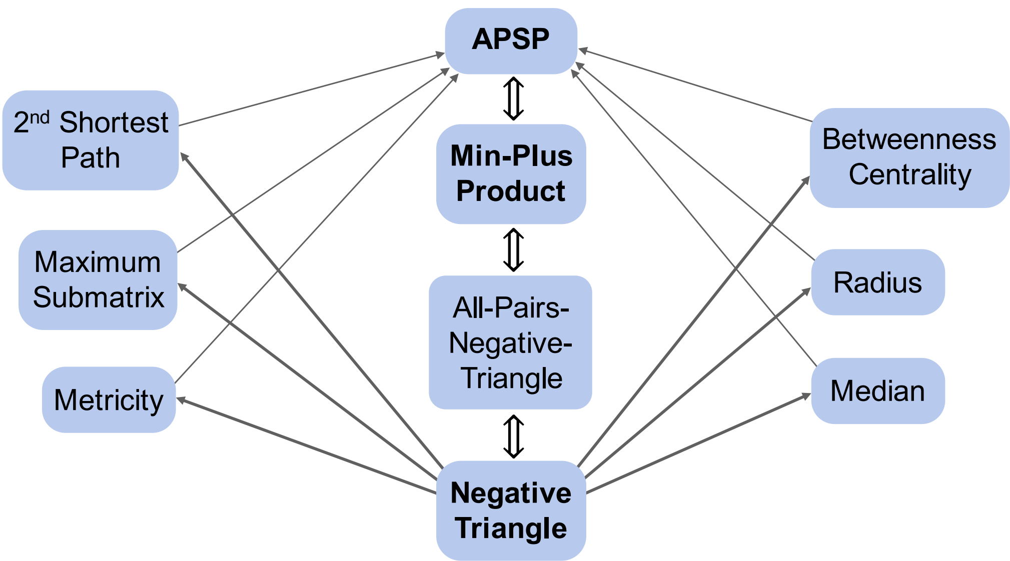
APSP and Min-Plus Product are **subcubic equivalent**

Cor: APSP has an $O(n^{3-\varepsilon})$ algorithm for some $\varepsilon > 0$ if and only if Min-Plus Product has an $O(n^{3-\delta})$ algorithm for some $\delta > 0$

Cor: Min-Plus Product is in time $O\left(n^3 / 2^{\Omega(\log n)^{1/2}}\right)$

max planck institut informatik

Subcubic Equivalences



[Vassilevska-Williams,Williams'10]

[Abboud,Grandoni,Vassilevska-Williams'15]

max planck institut informatik

Triangle Problems

Negative Triangle

each edge has a weight in $\{-n^c, .., n^c\}$

Given a weighted directed graph G

Decide whether **there are vertices i, j, k** such that

$$w(j,i) + w(i,k) + w(k,j) < 0$$

from definition: $O(n^3)$

no $O(n^{3-\varepsilon})$ algorithm known (which works for all $c > 0$)

Intermediate problem:

All-Pairs-Negative-Triangle

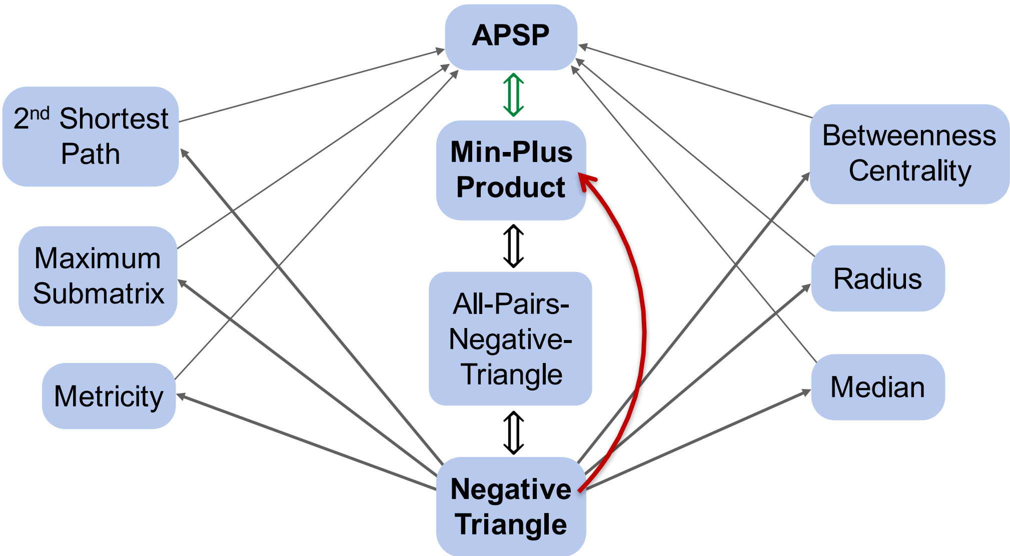
Given a weighted directed graph G with vertex set $V = I \cup J \cup K$

Decide **for every $i \in I, j \in J$ whether there is a vertex $k \in K$** s.t.

$$w(j,i) + w(i,k) + w(k,j) < 0$$

max planck institut informatik

Subcubic Equivalences



[Vassilevska-Williams,Williams'10]
[Abboud,Grandoni,Vassilevska-Williams'15]

Neg-Triangle to Min-Plus-Product

Min-Plus Product ↑ Negative Triangle

Given a weighted directed graph G on vertex set $\{1,\dots,n\}$

Adjacency matrix A:

$A_{i,j}$ = weight of edge (i,j), or ∞ if the edge does not exist

1. Compute Min-Plus Product $B \coloneqq A * A$:

   $B_{i,j} = \min_k A_{i,k} + A_{k,j}$

2. Compute $\min_{i,j} A_{j,i} + B_{i,j}$

   this equals $\min_{i,j,k} A_{j,i} + A_{i,k} + A_{k,j}$

   i.e. the smallest weight of any triangle

   thus we solved Negative Triangle

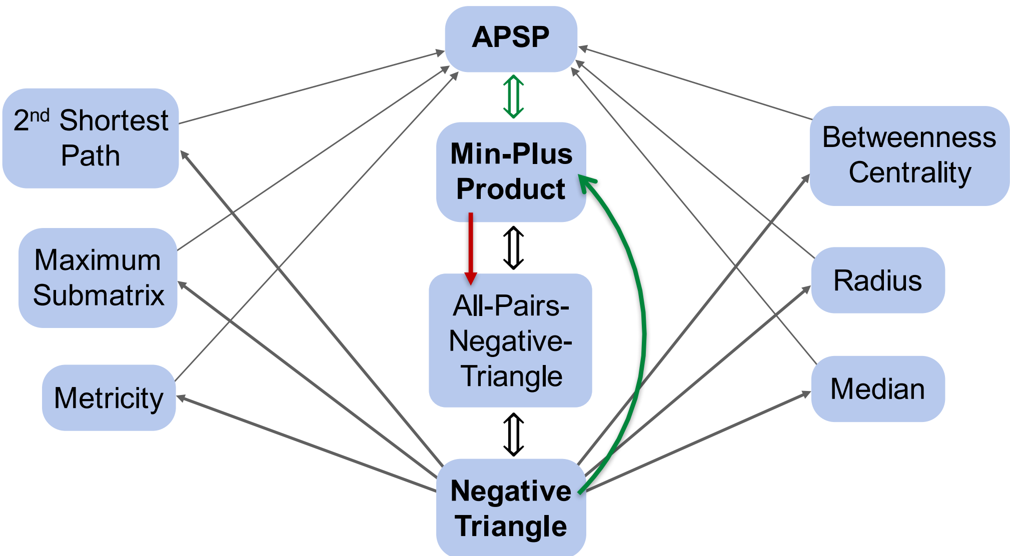
A:

3	1	∞	∞
∞	∞	4	∞
1	5	∞	2
2	∞	7	1

Running Time: $T_{\text{NegTriangle}}(n) \leq T_{\text{MinPlus}}(n) + O(n^2)$

→ subcubic reduction

Subcubic Equivalences



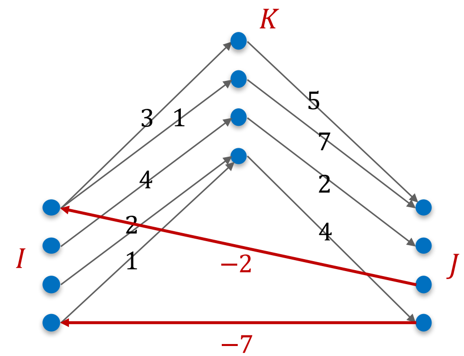
[Vassilevska-Williams,Williams'10]
[Abboud,Grandoni,Vassilevska-Williams'15]

Min-Plus to All-Pairs-Neg-Triangle

Min-Plus Product ↓ All-Pairs-Negative-Triangle

3	1	∞	∞
∞	∞	4	∞
∞	∞	∞	2
∞	∞	∞	1

A



5	∞	∞	∞
7	∞	∞	∞
∞	2	∞	∞
∞	∞	∞	4

B

$n = 4$ in the picture

Add all edges from J to I with (carefully chosen) weights $w(j,i)$

Run All-Pairs-Negative-Triangle algorithm

Result: for all i,j, is there a k such that $w(j,i) + w(i,k) + w(k,j) < 0$?

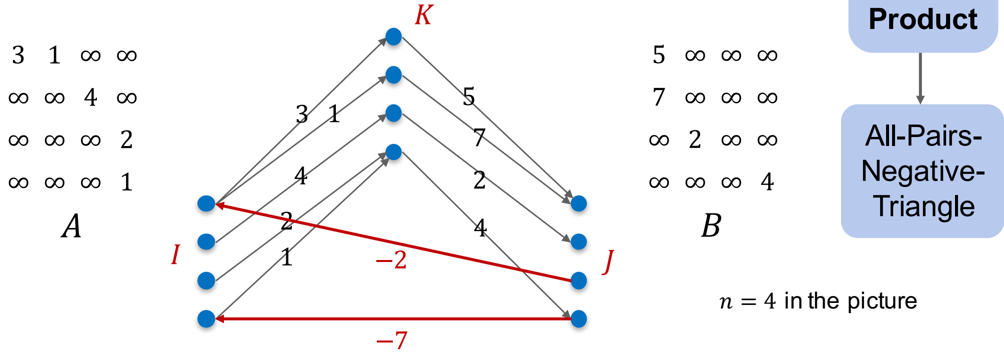
$\Leftrightarrow w(i,k) + w(k,j) < -w(j,i)$

WANTED: Min-Plus: for all i,j: $\min_k w(i,k) + w(k,j)$

= minimum number z s.t. there is a k s.t. $w(i,k) + w(k,j) < z + 1$

binary search via $w(j,i)$! **simultaneous** for all i,j!

Min-Plus to All-Pairs-Neg-Triangle



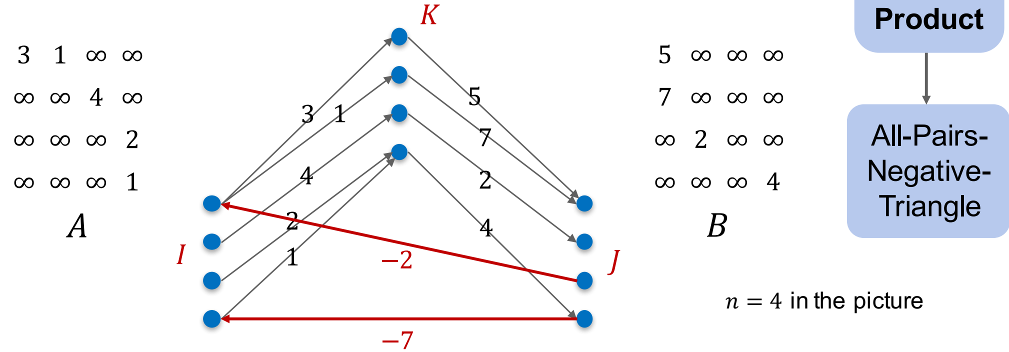
binary search via $w(j,i)$! **simultaneous** for all i,j!

need that all (finite) weights are in $\{-n^c, \dots, n^c\}$

each entry of Min-Plus Product is in $\{-2n^c, \dots, 2n^c, \infty\}$

binary search takes $\log_2(4n^c + 1) = O(\log n)$ steps

Min-Plus to All-Pairs-Neg-Triangle



binary search via $w(j,i)$! **simultaneous** for all i,j!

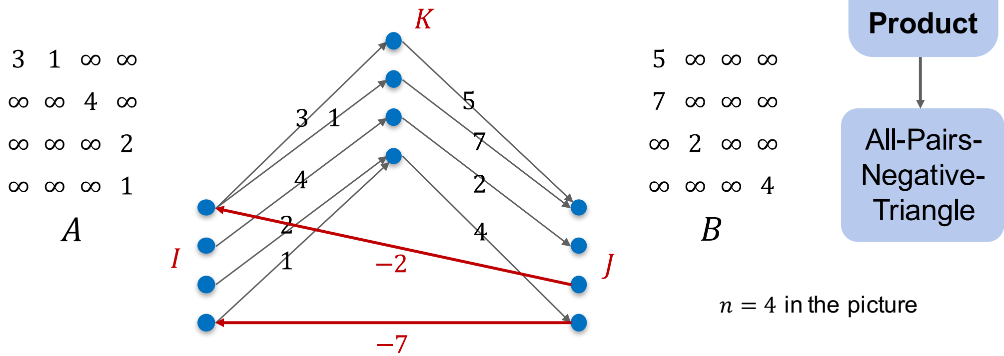
for all i,j: initialize $m(i,j) := -2n^c$ and $M(i,j) := 2n^c$

repeat $\log(4n^c)$ times:

- for all i,j: set $w(j,i) := -\lceil (m(i,j) + M(i,j))/2 \rceil$
- compute All-Pairs-Negative-Triangle
- for all i,j: if i,j is in negative triangle: $M(i,j) := -w(j,i) - 1$
  - otherwise: $m(i,j) := -w(j,i)$

(missing: handling of ∞)

Min-Plus to All-Pairs-Neg-Triangle



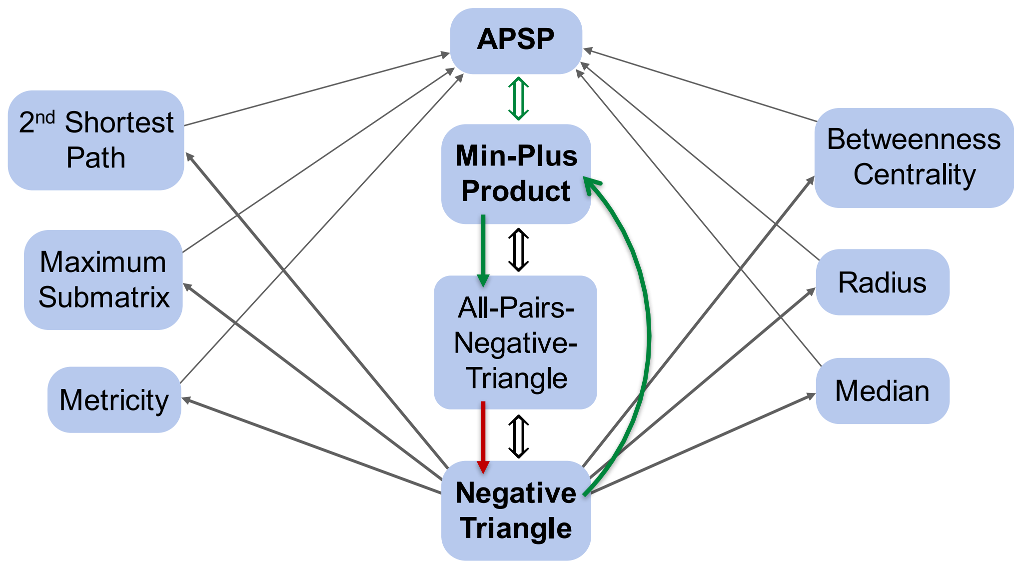
binary search takes $\log_2(4n^c + 1) = O(\log n)$ steps

$T(n)$ algorithm for All-Pairs-Neg-Triangle yields
$O(T(n) \log n)$ algorithm for Min-Plus Product

In particular: $O(n^{3-\varepsilon})$ algorithm for All-Pairs-Neg-Triangle for some $\varepsilon > 0$ implies $O(n^{3-\varepsilon})$ algorithm for Min-Plus Product for some $\varepsilon > 0$

→ subcubic reduction

Subcubic Equivalences



[Vassilevska-Williams,Williams'10]
[Abboud,Grandoni,Vassilevska-Williams'15]

All-Pairs-Neg-Triangle to Neg-Triangle

All-Pairs-Negative-Triangle → Negative Triangle

Negative Triangle Given graph G

Decide whether there are vertices i, j, k such that

$$w(j,i) + w(i,k) + w(k,j) < 0$$

All-Pairs-Negative-Triangle Given graph G with vertex set $V = I \cup J \cup K$

Decide for every $i \in I, j \in J$ whether there is a vertex $k \in K$ such that

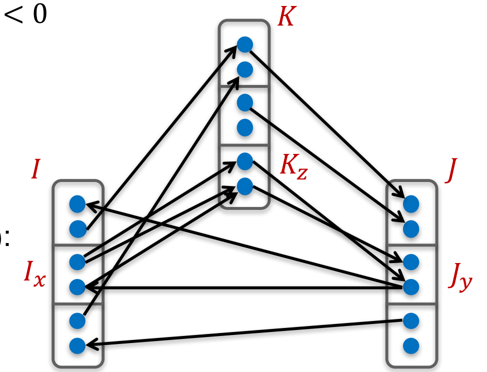
$$w(j,i) + w(i,k) + w(k,j) < 0$$

Split I, J, K into n/s parts of size s:

$I_1, \ldots, I_{n/s}, J_1, \ldots, J_{n/s}, K_1, \ldots, K_{n/s}$

For each of the $(n/s)^3$ triples (I_x, J_y, K_z):

consider graph $G[I_x \cup J_y \cup K_z]$



All-Pairs-Neg-Triangle to Neg-Triangle

All-Pairs-Negative-Triangle → Negative Triangle

Initialize C as $n \times n$ all-zeroes matrix

For each of the $(n/s)^3$ triples of parts (I_x, J_y, K_z):

While $G[I_x \cup J_y \cup K_z]$ contains a negative triangle:

Find a negative triangle (i,j,k) in $G[I_x \cup J_y \cup K_z]$

Set $C[i,j] := 1$

Set $w(i,j) := \infty$

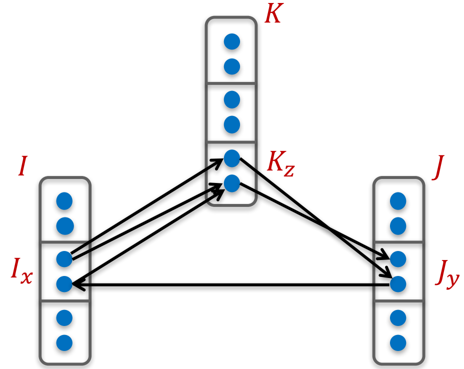
(i,j) is in no more negative triangles

✔ guaranteed termination:

can set $\leq n^2$ weights to ∞

✔ correctness:

if (i,j) is in negative triangle,
we will find one



All-Pairs-Neg-Triangle to Neg-Triangle

All-Pairs-Negative-Triangle → Negative Triangle

Find a negative triangle (i,j,k) in $G[I_x \cup J_y \cup K_z]$

*How to **find** a negative triangle*
*if we can only **decide** whether one exists?*

Partition I_x into $I_x^{(1)}, I_x^{(2)}$, J_y into $J_y^{(1)}, J_y^{(2)}$, K_z into $K_z^{(1)}, K_z^{(2)}$

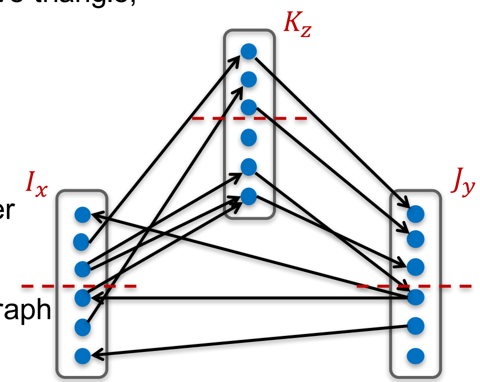
Since $G[I_x \cup J_y \cup K_z]$ contains a negative triangle,
at least one of the 2^3 subgraphs

$$G[I_x^{(a)} \cup J_y^{(b)} \cup K_z^{(c)}]$$

contains a negative triangle

Decide for each such subgraph whether
it contains a negative triangle

Recursively find a triangle in one subgraph



All-Pairs-Neg-Triangle to Neg-Triangle

All-Pairs-Negative-Triangle → Negative Triangle

Find a negative triangle (i,j,k) in $G[I_x \cup J_y \cup K_z]$

*How to **find** a negative triangle*
*if we can only **decide** whether one exists?*

Partition I_x into $I_x^{(1)}, I_x^{(2)}$, J_y into $J_y^{(1)}, J_y^{(2)}$, K_z into $K_z^{(1)}, K_z^{(2)}$

Since $G[I_x \cup J_y \cup K_z]$ contains a negative triangle,
at least one of the 2^3 subgraphs

$$G[I_x^{(a)} \cup J_y^{(b)} \cup K_z^{(c)}]$$

contains a negative triangle

Decide for each such subgraph whether
it contains a negative triangle

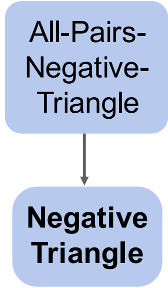
Recursively find a triangle in one subgraph

Running Time:

$$T_{\text{FindNegTriangle}}(n) \leq 2^3 \cdot T_{\text{DecideNegTriangle}}(n) + T_{\text{FindNegTriangle}}(n/2)$$

$$= O(T_{\text{DecideNegTriangle}}(n))$$

All-Pairs-Neg-Triangle to Neg-Triangle



Initialize C as $n \times n$ all-zeroes matrix

For each of the $(n/s)^3$ triples of parts (I_x, J_y, K_z):

While $G[I_x \cup J_y \cup K_z]$ contains a negative triangle:

Find a negative triangle (i,j,k) in $G[I_x \cup J_y \cup K_z]$

Set $C[i,j] := 1$

Set $w(i,j) := \infty$

$(*)$

Running Time:

$(*) = O(T_{\text{FindNegTriangle}}(s)) = O(T_{\text{DecideNegTriangle}}(s))$

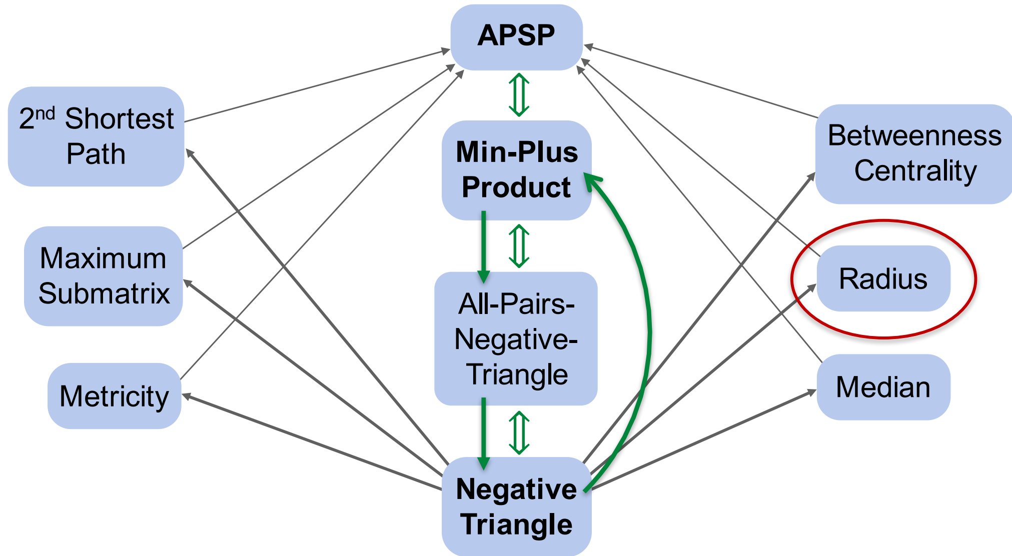
Total time: $\big((\#\text{triples}) + (\#\text{triangles found})\big) \cdot (*)$

$\leq \big((n/s)^3 + n^2\big) \cdot T_{\text{DecideNegTriangle}}(s)$

Set $s = n^{1/3}$ and assume $T_{\text{DecideNegTriangle}}(n) = O(n^{3-\varepsilon})$

Total time: $O\big(n^2 \cdot n^{1-\varepsilon/3}\big) = O(n^{3-\varepsilon/3})$

Subcubic Equivalences

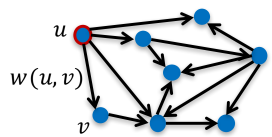


[Vassilevska-Williams,Williams'10]
[Abboud,Grandoni,Vassilevska-Williams'15]

Radius

G is a weighted directed graph

$d(u,v)$ is the distance from u to v in G



Radius: $\min_u \max_v d(u,v)$

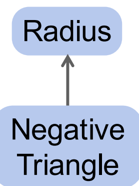
u is in some sense the *most central vertex*

compute all pairwise distances,
then evaluate definition of radius in time $O(n^2)$

→ subcubic reduction

$\Rightarrow$ Radius is in time $O\left(n^3 / 2^{\Omega(\log n)^{1/2}}\right)$

Negative Triangle to Radius

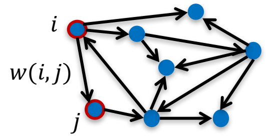


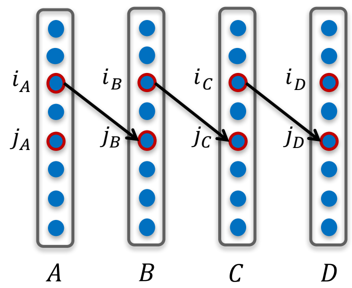
Negative Triangle instance:
graph G with n nodes,
edge-weights in $\{-n^c, \ldots, n^c\}$

→

Radius instance:
graph H with $O(n)$ nodes,
edge-weights in $\{0, \ldots, O(n^c)\}$

$M := 3n^c$





1) Make four layers with n nodes

2) For any edge (i,j): Add (i_A, j_B), (i_B, j_C), (i_C, j_D) with weight $M + w(i,j)$

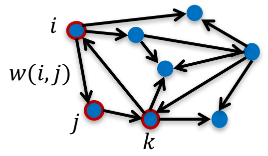
Negative Triangle to Radius

Radius ← Negative Triangle

Negative Triangle instance:
graph G with n nodes,
edge-weights in $\{-n^c, \ldots, n^c\}$

→

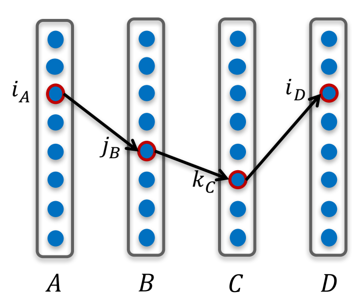
Radius instance:
graph H with $\mathrm{O}(n)$ nodes,
edge-weights in $\{0, \ldots, O(n^c)\}$



(i, j, k) has weight W

1) Make four layers with n nodes
2) For any edge (i, j): Add (i_A, j_B), (i_B, j_C), (i_C, j_D) with weight $M + w(i, j)$

$M := 3n^c$



⇔ path has length $3M + W$

→ $\exists i_A, j_B, k_C, i_D$-path of length $\leq 3M - 1$?

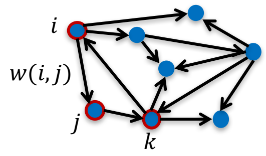
Negative Triangle to Radius

Radius ← Negative Triangle

Negative Triangle instance:
graph G with n nodes,
edge-weights in $\{-n^c, \ldots, n^c\}$

→

Radius instance:
graph H with $\mathrm{O}(n)$ nodes,
edge-weights in $\{0, \ldots, O(n^c)\}$

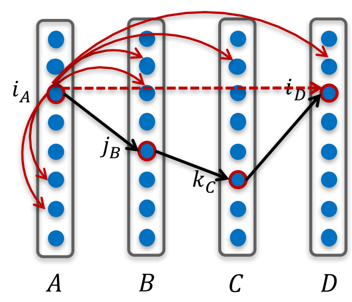


(i, j, k) has weight W

1) Make four layers with n nodes
2) For any edge (i, j): Add (i_A, j_B), (i_B, j_C), (i_C, j_D) with weight $M + w(i, j)$
3) Add edges of weight $3M - 1$ from any i_A to all nodes except i_D (and i_A)

Radius: $\min_u \max_v d(u, v)$

$M := 3n^c$



⇔ path has length $3M + W$

→ $\exists i_A, j_B, k_C, i_D$-path of length $\leq 3M - 1$?

Claim: Radius of H is $\leq 3M - 1$ iff there is a negative triangle in G

Negative Triangle to Radius

Radius ← Negative Triangle

Claim: Radius of H is $\leq 3M - 1$ iff there is a negative triangle in G

Proof:

If there is a negative triangle (i, j, k) then i_A is in distance $\leq 3M - 1$ to i_D (by (2)), and in distance $\leq 3M - 1$ to any other vertex (by (3)),
so the radius is $\leq \max_v d(i_A, v) \leq 3M - 1$

If there is no negative triangle (i, j, k):
Any node u of the form $i_B/i_C/i_D$ cannot reach A, so it has $\max_v d(u, v) = \infty$
Any i_A is in distance $\geq 3M$ to i_D, since there is no i_A, j_B, k_C, i_D-path of length $\leq 3M - 1$ (note that the edges added in (3) also do not help)
Hence, for all u, $\max_v d(u, v) \geq 3M$, and thus the radius is at least $3M$