LA SAPIENZA

Maintaining Shortest Paths in Digraphs with Arbitrary Arc Weights

C. Demetrescu
D. Frigioni
A. Marchetti-Spaccamela
U. Nanni

University of Rome "La Sapienza"

Fully Dynamic Single-source Shortest Paths

Let: $G = (V,E,w)$ weighted directed graph

$w(u,v) \in \Re$ weight of edge (u,v)

$s \in V$ source node

Perform intermixed sequence of operations:

`Increase(u,v,ε):`	Increase weight $w(u,v)$ by ε
`Decrease(u,v,ε):`	Decrease weight $w(u,v)$ by ε
`Query(v):`	Return shortest path from s to v in G

A Simple-minded Method

After each `Increase` or `Decrease`:

use best static algorithm [Bellman-Ford]

to recompute from scratch shortest paths in G

$\rightarrow O(m \cdot n)$ worst-case time ($n=|V|$, $m=|E|$)

Can we do any better?

An Asymptotically Faster Method

After each `Increase` or `Decrease`:

1. Use a reweighting technique to obtain from G a new graph G* with nonnegative weights $\rightarrow O(m)$
2. Perform Dijkstra's algorithm to compute from scratch shortest paths in G* $\rightarrow O(m+n \cdot \log n)$
3. Retrieve shortest paths in G from shortest paths in G* $\rightarrow O(n)$

Thus: $O(m \cdot n) \Rightarrow O(m+n \cdot \log n)$

A Reweighting Technique [Edmonds, Karp]

$G = (V,E,w)$ $\qquad w : E \rightarrow \Re$

Reweighting

$$G^* = (V,E,w^*) \quad \begin{cases} h : V \rightarrow \Re \quad \text{(arbitrary)} \\ w^*(u,v) = w(u,v) + h(u) - h(v) \end{cases}$$

Lemma 1:

p is a shortest path in $G \Leftrightarrow p$ is a shortest path in G^*

A Reweighting Technique
[Ramalingam and Reps]

If we choose:

$h(v) := d(v) =$ distance from s to v

Lemma 2:

$w^*(u,v) = w(u,v) + d(u) - d(v) \geq 0$

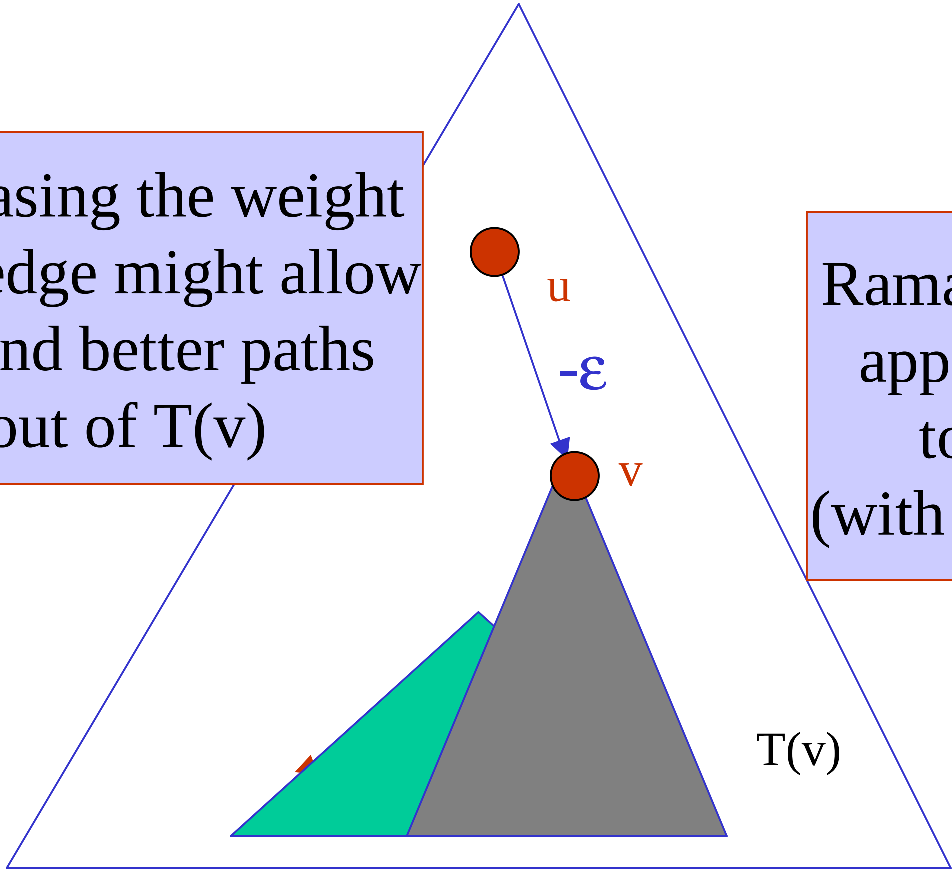
Proof:

$d(v) \leq w(u,v) + d(u)$ [Bellman]

$0 \leq w(u,v) + d(u) - d(v)$

Weight decrease [Ramalingam and Reps]

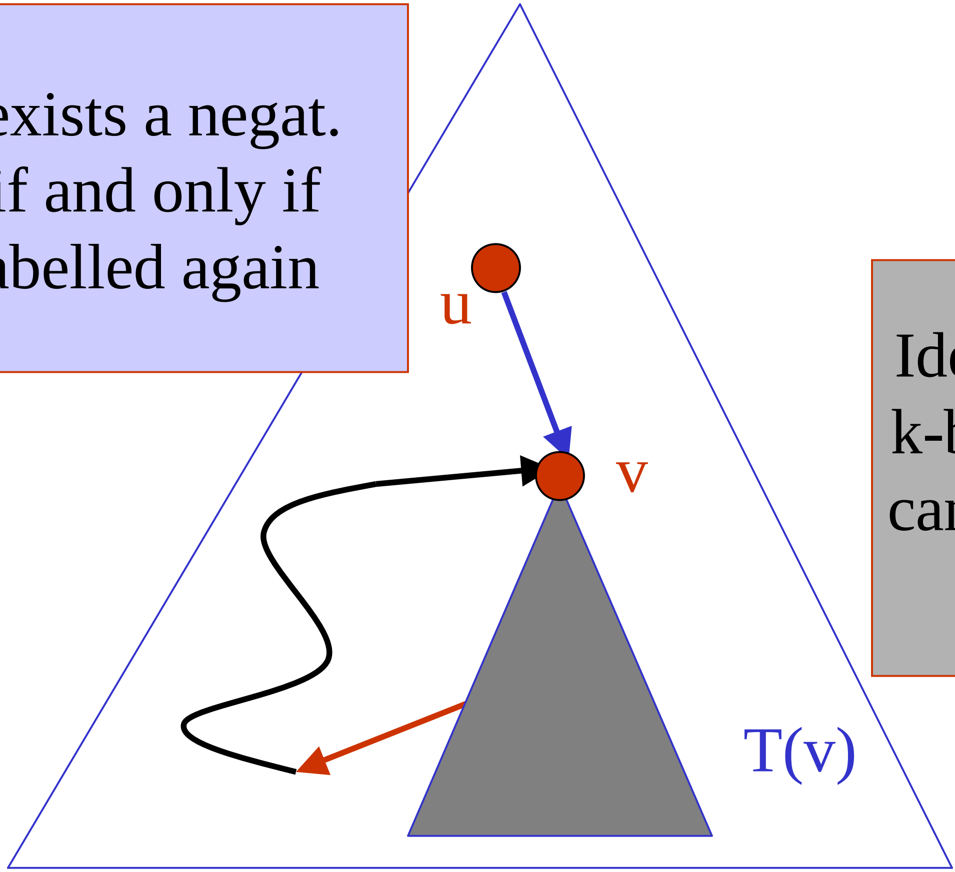
Decreasing the weight of an edge might allow to find better paths out of T(v)



Ramalingam and Reps: apply Dijkstra's alg. to the graph G* (with modified weights)

Weight decrease (cont.)

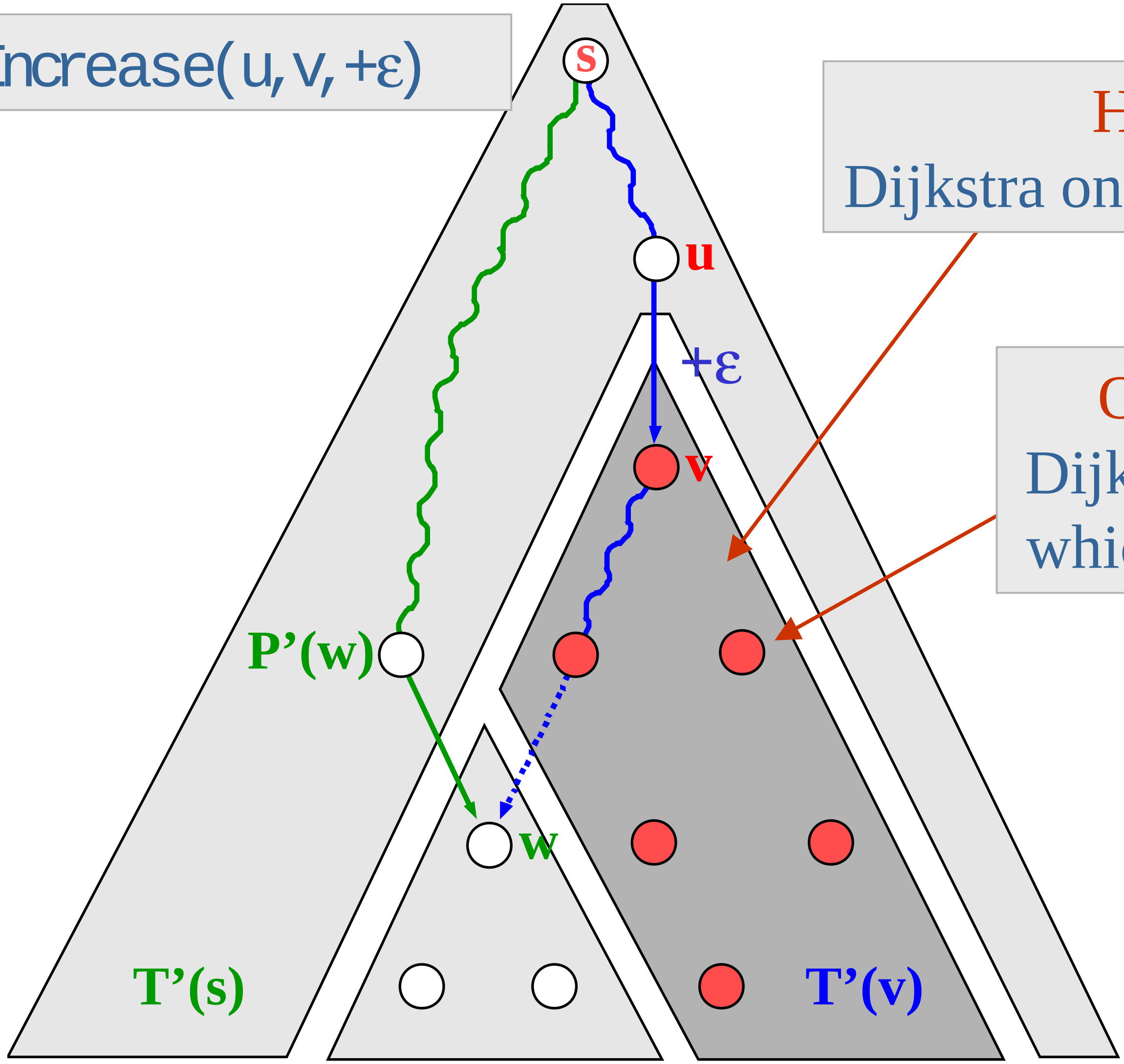
There exists a negat. cycle if and only if v is labelled again



Ideas of ownership and k-bounded account. fct. can be applied reducing w.c.running time

Weight increase: Output Bounded Analysis

Increase(u,v,+ε)



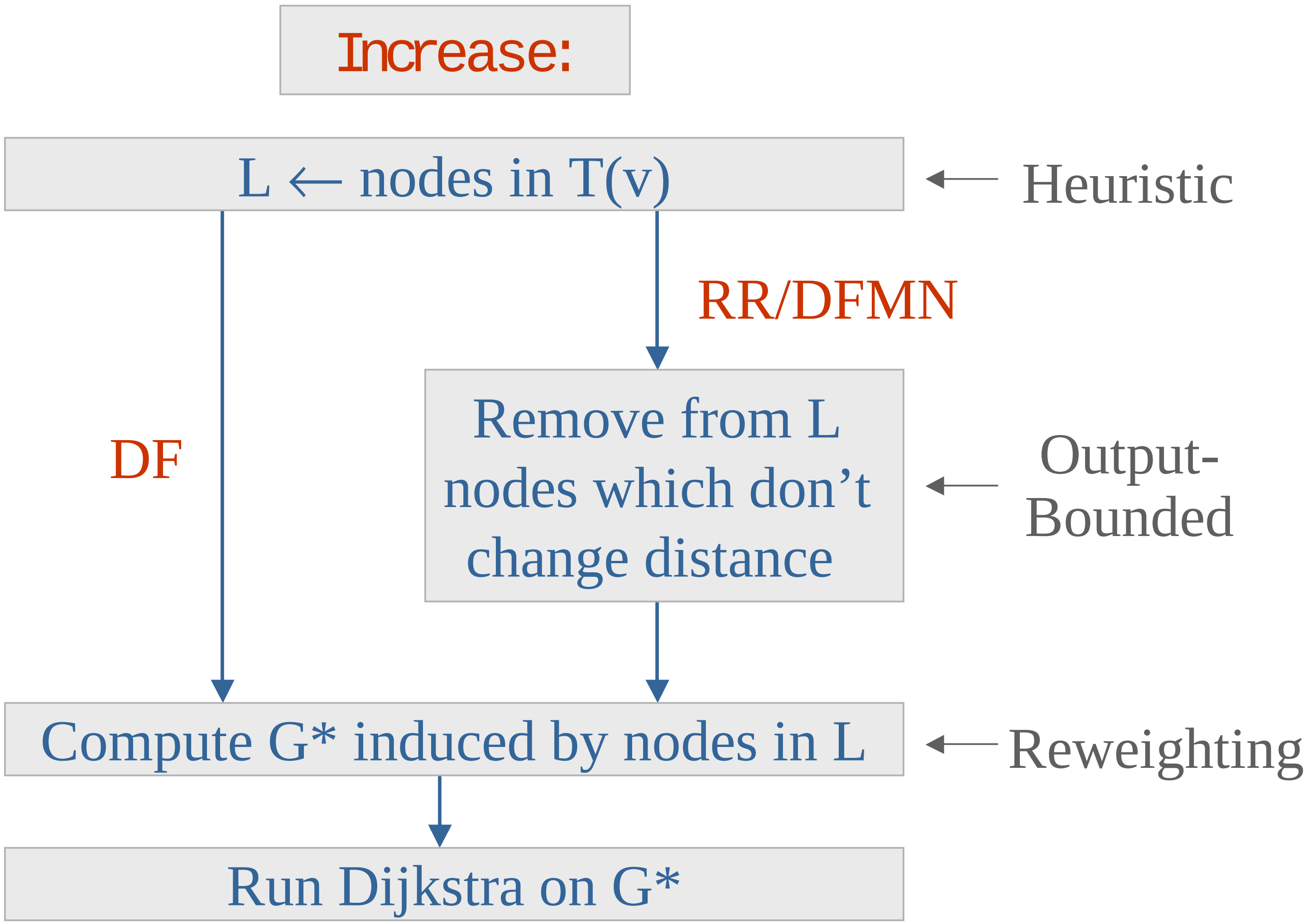
Heuristic:
Dijkstra only on nodes in T(v)

Output bounded:
Dijkstra only on nodes which change distance

Algorithms Under Evaluation

Name	Technique	Update time
BF	Simple-minded method [Bellman-Ford]	$O(m \cdot n)$
DF	Reweighting method + Heuristic	$O(m+n \cdot \log n)$
RR	Reweighting method + Output Bounded (Does not deal with zero length cycles) [Ramalingam, Reps]	$O(m+n \cdot \log n)$
DFMN	Reweighting method + Output Bounded [Frigioni, Marchetti, Nanni]	$O(m+n \cdot \log n)$

DF vs RR/DFMN

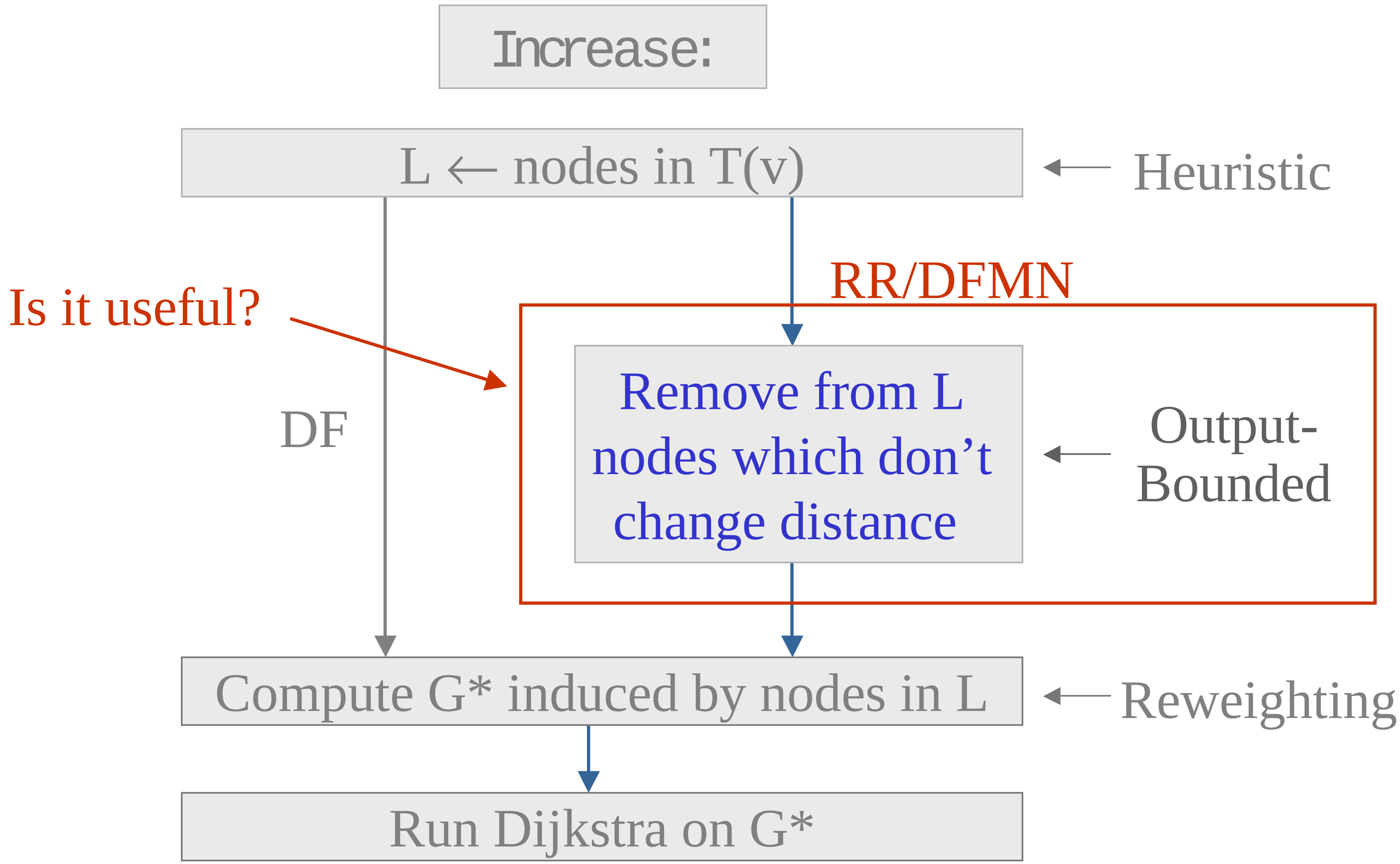

Increase:
L ← nodes in T(v)
Heuristic
RR/DFMN
DF
Remove from L nodes which don't change distance
Output-Bounded
Compute G* induced by nodes in L
Reweighting
Run Dijkstra on G*

Goals of Experimentation

Look for hints about questions like:

1	We know that $O(m+n\cdot\log n)$ is better than $O(m\cdot n)$ … … but what about constant factors?
2	DFMN and RR are efficient in output-bounded complexity … …but is it useful in practice?

DF vs RR/DFMN


Increase:
L ← nodes in T(v)
Heuristic
RR/DFMN
Is it useful?
DF
Remove from L nodes which don’t change distance
Output-Bounded
Compute G* induced by nodes in L
Reweighting
Run Dijkstra on G*

Experimental Setup

Experimental platform:

- C++ using LEDA, g++ compiler
- UNIX Solaris on SPARC Ultra 10 at 300 Mhz

Test sets:

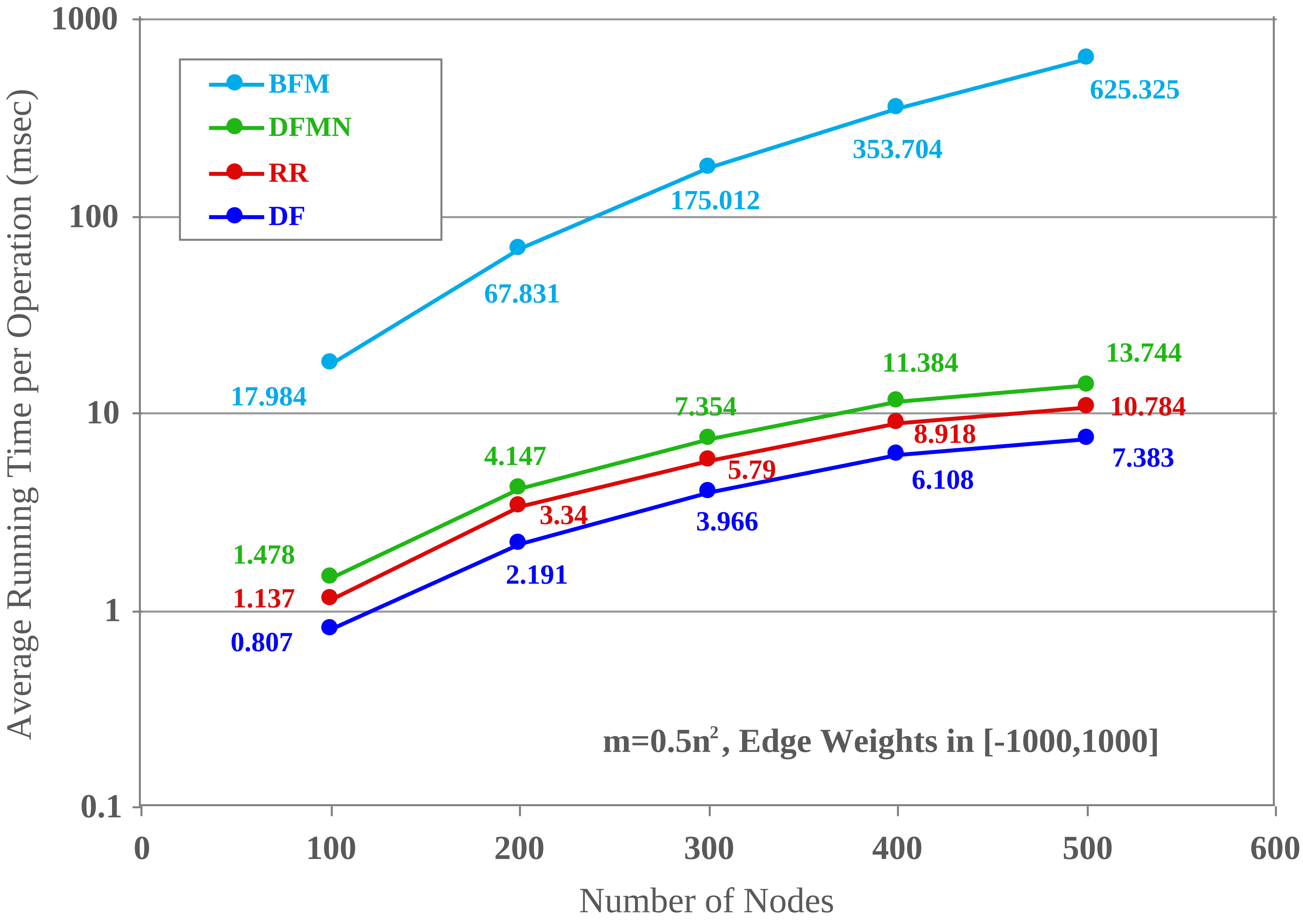
- Random graphs & random update sequences

  (Use potentials technique to avoid negative and zero-length cycles)

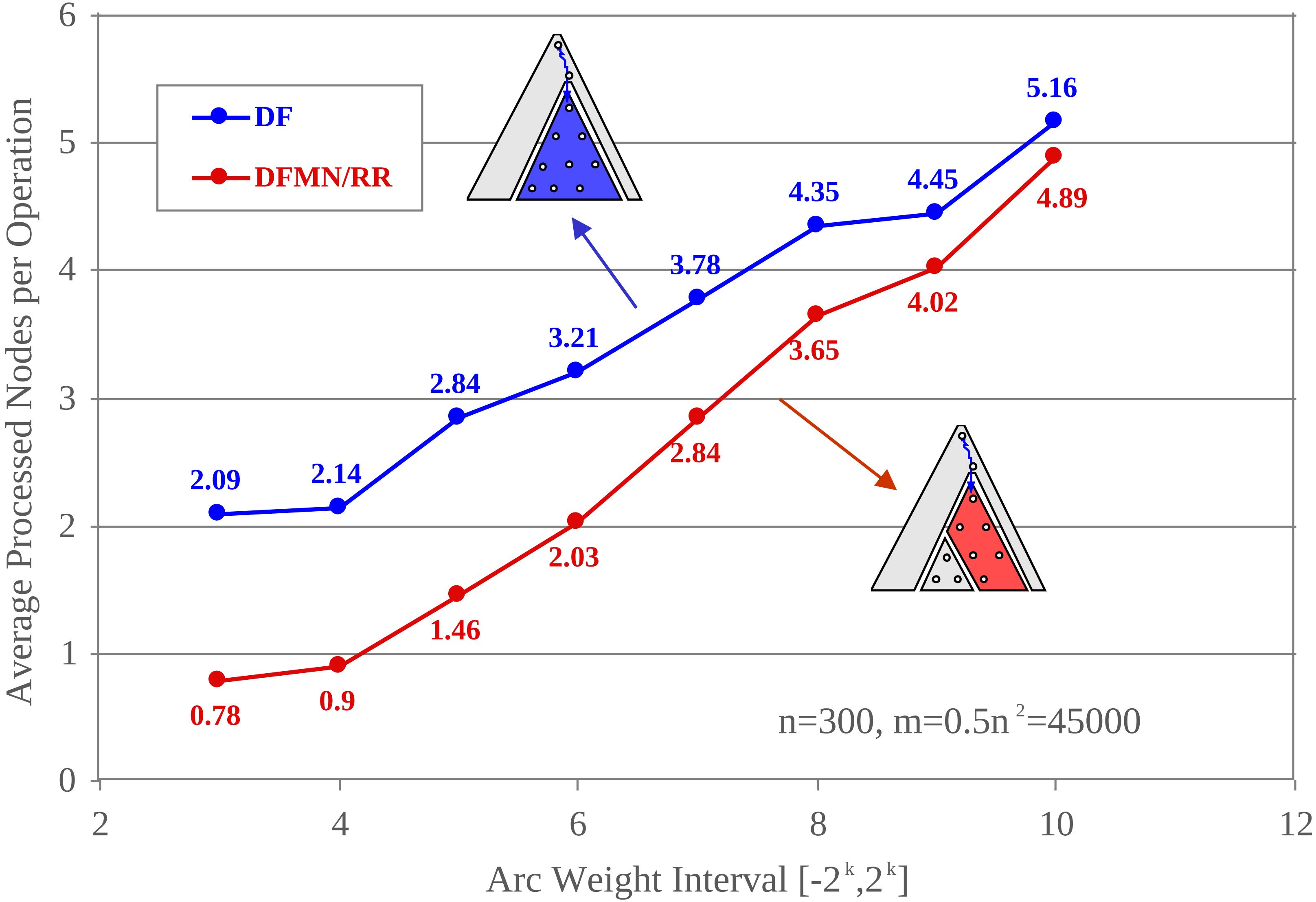
Performance indicators:

- Running time (msec)
- # nodes processed by Dijkstra's algorithm

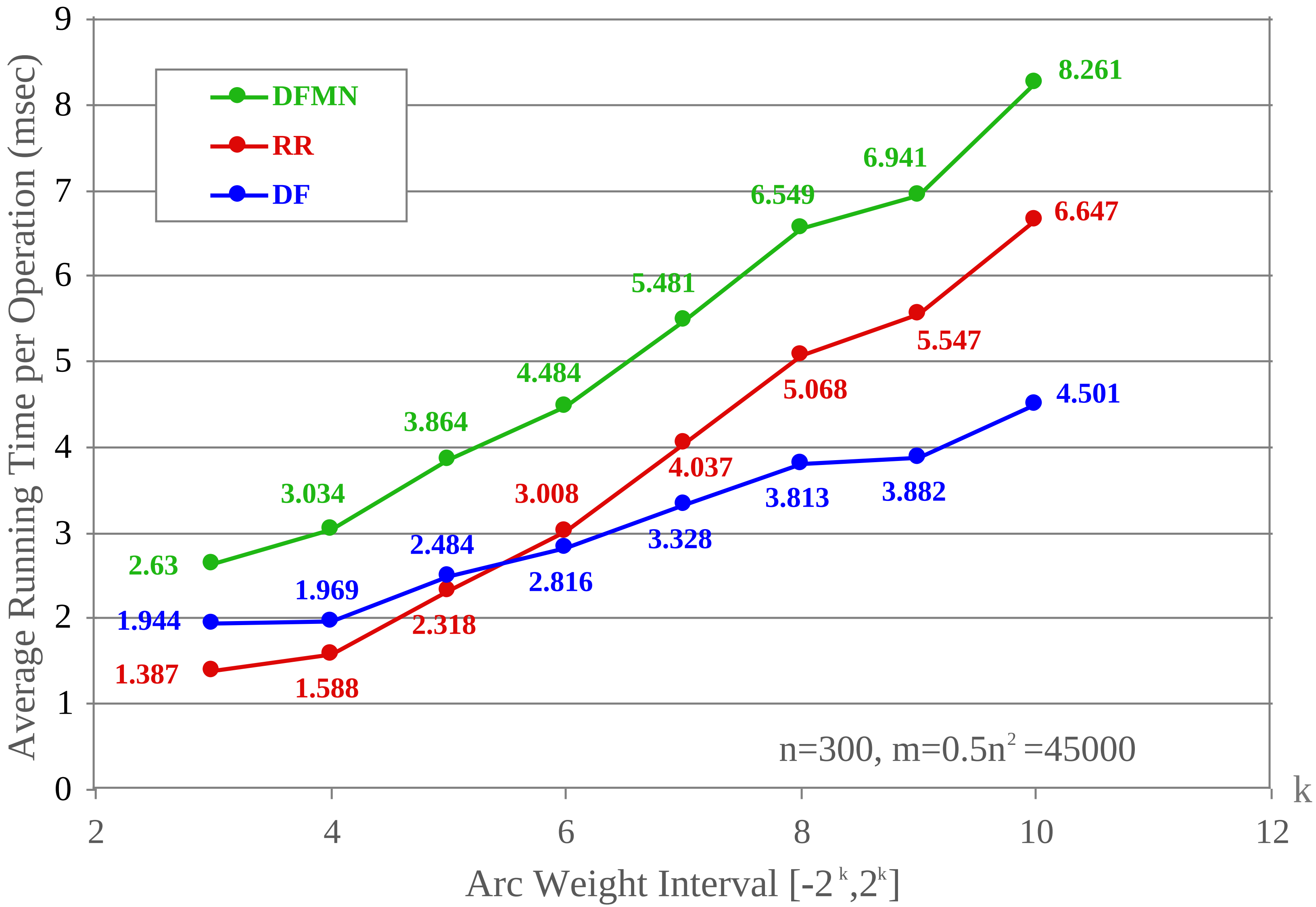
Constant Factors Are Small



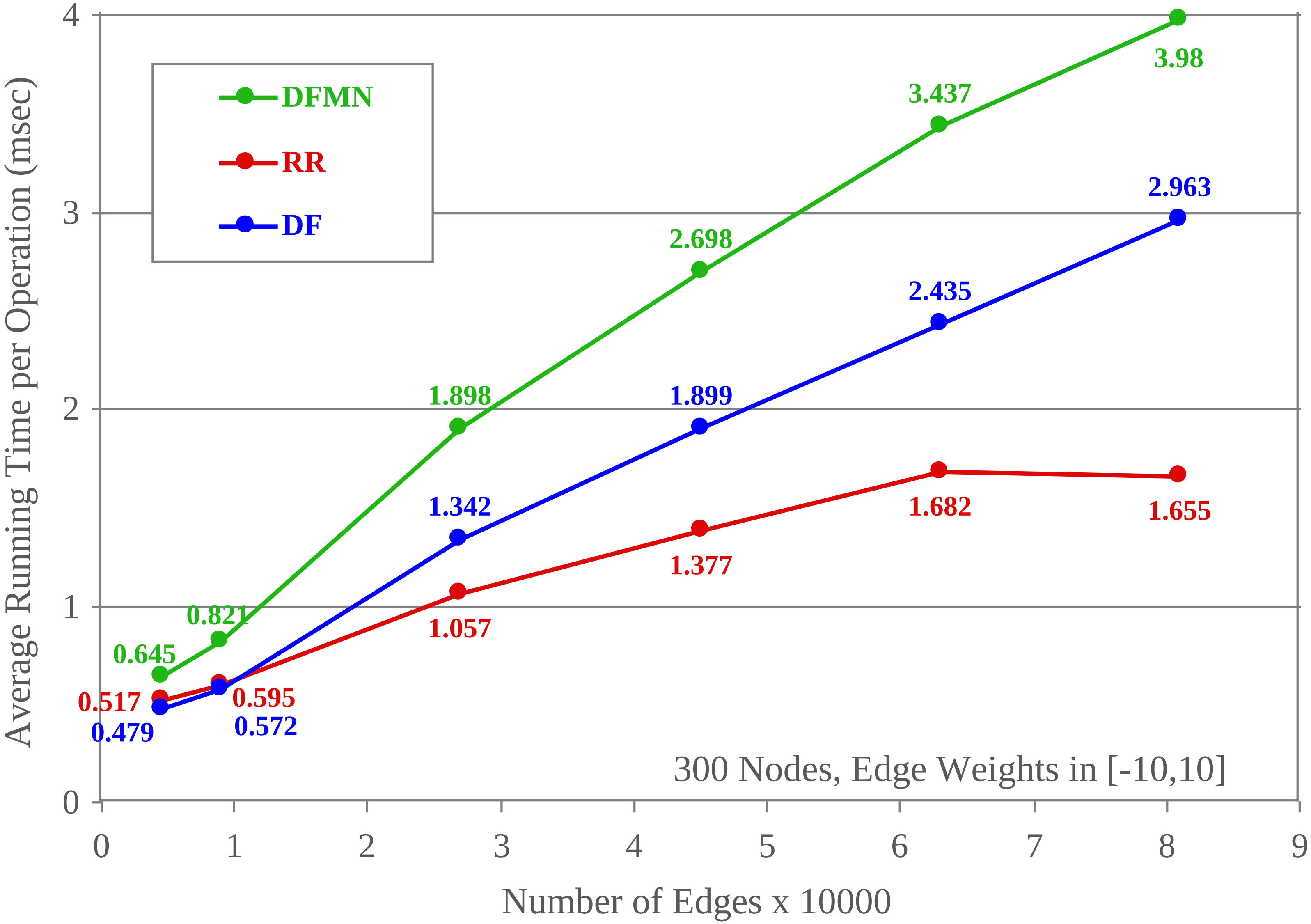
Nodes Processed by Dijkstra's Algo



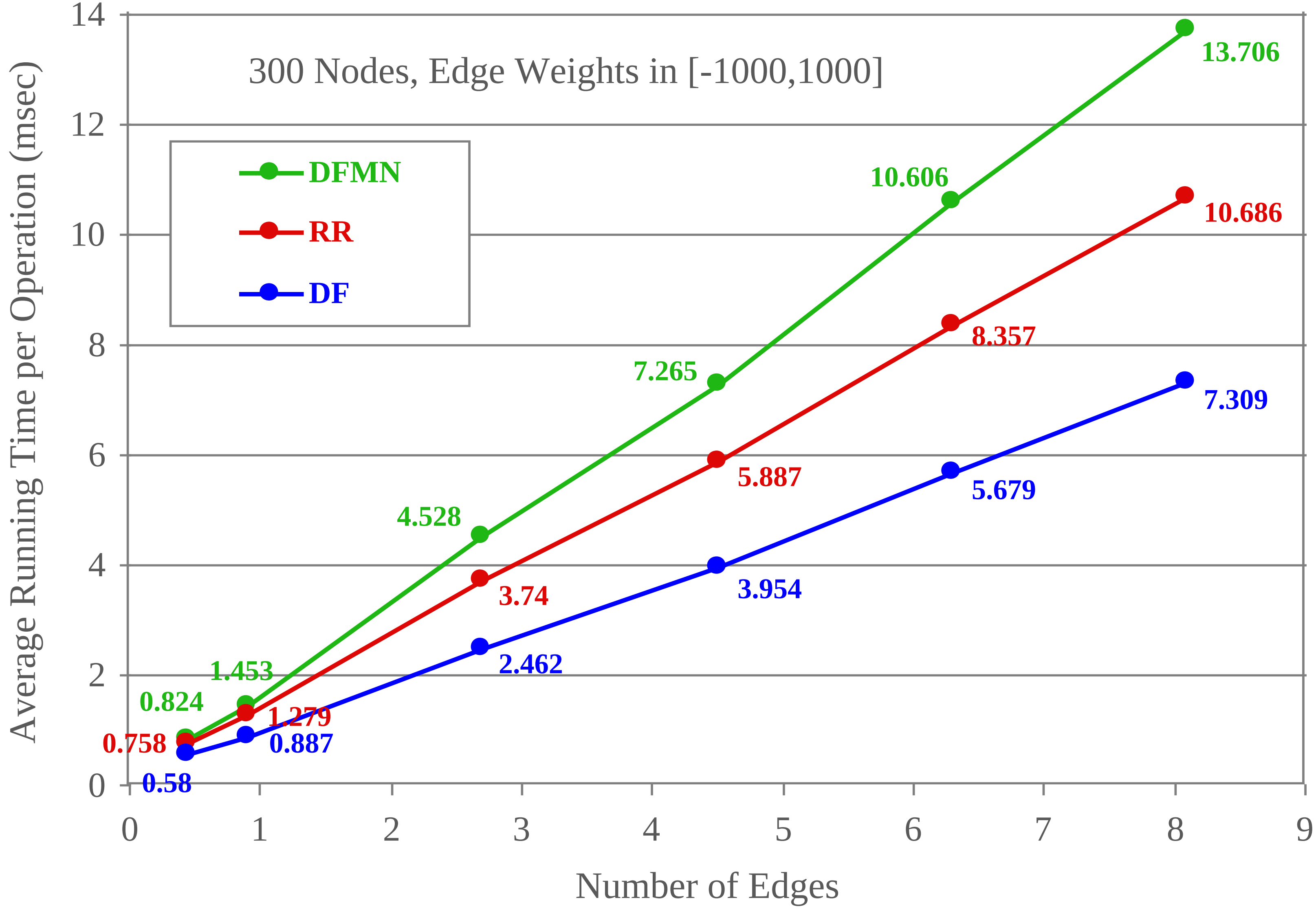
The Range of Arc Weights Is Important



Small Arc Weights Range



Large Arc Weights Range



Extreme Case: All Zero-length Cycles...

Conclusions

Dynamic algorithms based on the reweighting technique are very useful

In general, the simpler, the faster

Output bounded is useful for small ranges of arc weights

What happens on real test sets?

What happens on larger test sets?

Algorithms in a Nutshell (Increase)

BF	Bellman-Ford on G	
DF	L ← nodes in T(v)	Heuristic
	Compute G* induced by nodes in L	
	Run Dijkstra on G*	
RR DFMN	L ← nodes in T(v)	Heuristic
	Remove from L nodes which don't change distance	Output Bounded
	Compute G* induced by nodes in L	
	Run Dijkstra on G*	