Towards an Efficient Weighted Random Walk Domination

Songsong Mo, Zhifeng Bao, Ping Zhang, Zhiyong Peng

Outline

- Problem Definition
- Applications
- Solutions
- Experiments

Problem Definition: **Weighted Random Walk Model**

Probability-aware Random Walk Model (PR)

Edge weight

the probability that a node chooses or affects its neighbors

Cost-aware Random Walk Model (CR)

Edge weight

the cost of moving a node to its neighbors

Problem Definition

The expected cost for *u* to hit *S* via weighted random walk:

$$C_{uS} = \begin{cases} \sum_{w_{uS} \in \mathcal{W}_{uS}} |w_{uS}| \mathcal{P}(w_{uS}) & \text{Probability-aware Random Walk Model} \\ \sum_{w_{uS} \in \mathcal{W}_{uS}} c(w_{uS}) \mathcal{P}(w_{uS}) & \text{Cost-aware Random Walk Model} \end{cases}$$

C_{uS}^{B} is used to denote the value of C_{uS} bounded by budget *B*.

Weighted Random Walk Domination (WRWD)

- **Input:** Weighted graph *G*(*V*,*E*); Budget *B*.
- **Output:** A *k*-size set $S \subseteq V$ that can maximize the budget *B* to be saved:

$$\mathcal{G}(S) = \sum_{u \in V} \left(B - C_{uS}^{\mathrm{B}}\right)$$

Monotonicity & Submodularity

Outline

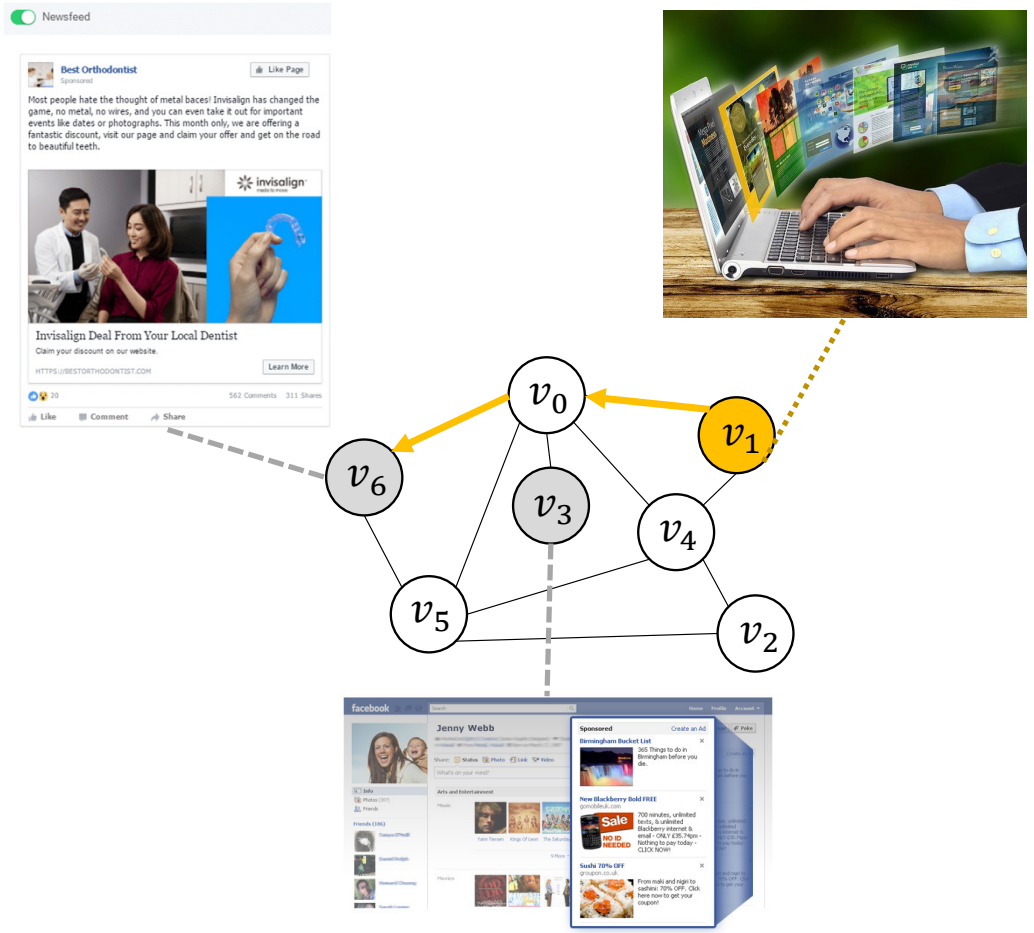
- Problem Definition
- Applications
- Solutions
- Experiments

Applications

Social Browsing

Page Browsing Process

Probability-aware Random Walk
(length ≤ budget B)



how to select pages in the network so that other users can easily browse these pages?

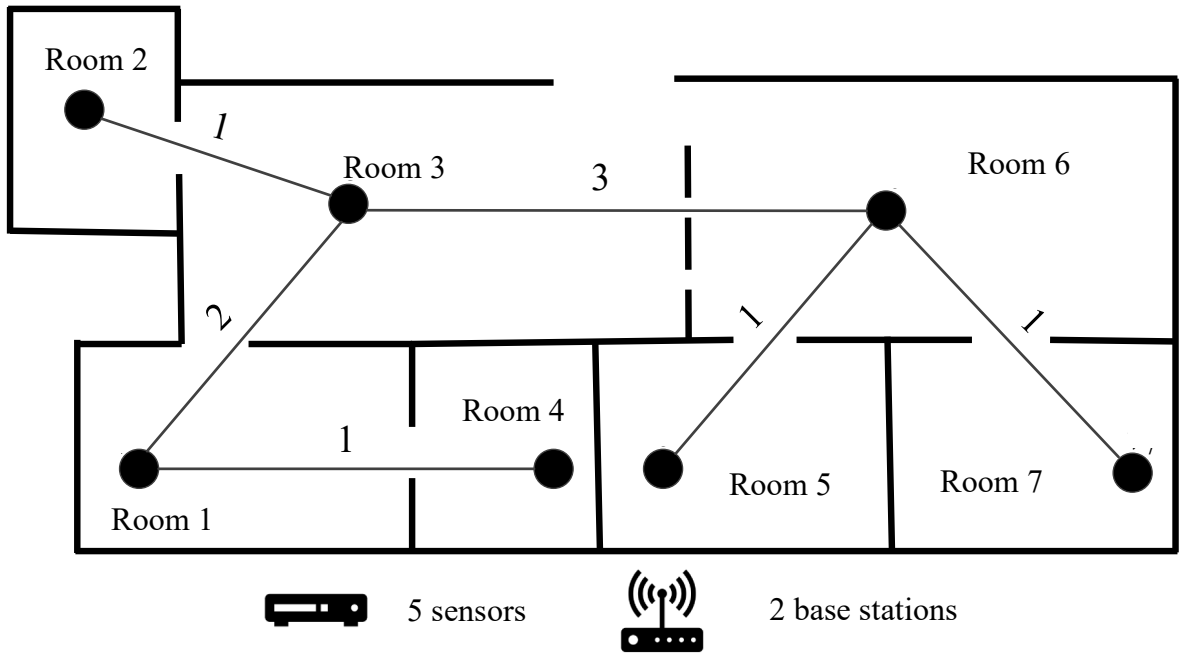
Applications

Base Station Selection

Data Transmission

Cost-aware Random Walk
(energy cost ≤ budget *B*)

How to choose a node set as base stations to minimize the energy cost?



Outline

- Problem Definition
- Applications
- Solutions
- Experiments

Solutions: An Overview

	Apply to P-WRWD		Apply to C-WRWD	
Algorithm	Time Complexity	Guarantee	Time Complexity	Guarantee
DpSel	$O(kBn^2m)$	$1-1/e$	$O(\delta kBn^2m)$	$1-1/e$
MatrixSel	$O(kBnm)$		$O(\delta kBnm)$	
BoundSel	$O(kBnm)$		$O(\delta kBnm)$	

***P-WRWD*:** the WRWD over PR

***C-WRWD*:** the WRWD over CR

***δ*:** the average of the edge weight

Solutions: A Greedy Framework

K iterations: select the node with the largest marginal gain → approximation ratio: 1-1/*e*

↓

Challenge : how to compute C_{uS}^{B} effectively

Strategy :

- Dynamic programming
- Matrix-based solution

Solutions : DpSel

Dynamic-programming-based selection (DpSel)

Recursive manner: $C_{uS}^{\mathrm{B}} = \begin{cases} 0 & u \in S \\ 1 & u \notin S, B = 1 \\ 1 + \sum_{v \in V} p_{uv} C_{vS}^{\mathrm{B}-1} & u \notin S, B > 1 \end{cases}$

Time Complexity

Calculate marginal gain one time → Call DP $O(n)$ time → Time Complexity $O(Bnm)$

↓

Each iteration: select the node with the largest marginal gain → Time Complexity $O(Bn^2m)$

↓

K iterations → Time Complexity $O(kBn^2m)$

Too slow

Solutions : MatrixSel

Core idea : Calculate C_{uS}^{B} of all nodes at the same time by the matrix

Trapped Model

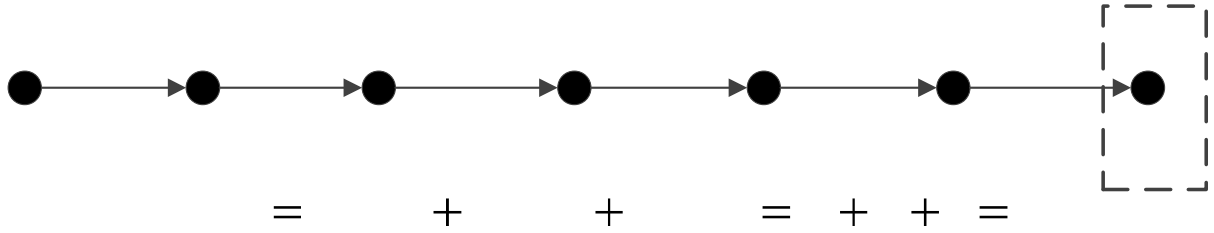
Any walk steps after S is visited will have no effect on C_{uS}^{B}

Transition matrix **P** —reconstruction→ $\begin{vmatrix} Q_{\bar{s}} & \mathbf{R} \\ \mathbf{0} & \mathbf{0}_s \end{vmatrix}$

$Q_{\bar{s}}$ represents the sub-transition matrix corresponding to $\bar{s}$

Solutions : MatrixSel

The bridge between C_{uS}^{B} and $Q_{\bar{s}}$



Observation: The length of random walk w_{uS} equals to $\sum_{v \in \bar{S}} Y(v)$, where $Y(v)$ denotes the frequency of w_{uS} reaching v

Lemma: The sum of the u-th row of $\mathbf{N}_{\bar{s}}$ is equal to C_{uS}^{B}

Computing $\mathbf{N}_{\bar{s}}\mathbf{I}$ to get C_{uS}^{B} of all nodes

$$\mathbf{N}_{\bar{s}} = \{\mathbf{Q}_{\bar{s}}^{1} + \mathbf{Q}_{\bar{s}}^{2} + \mathbf{Q}_{\bar{s}}^{3} + \ldots + \mathbf{Q}_{\bar{s}}^{B}\}$$

Solutions : MatrixSel

Computing $\mathbf{N}_{\bar{s}}\mathbf{I}$ $\mathbf{N}_{\bar{s}} = \{\mathbf{Q}_{\bar{s}}^1 + \mathbf{Q}_{\bar{s}}^2 + \mathbf{Q}_{\bar{s}}^3 + \ldots + \mathbf{Q}_{\bar{s}}^B\}$

Computing $\mathbf{Q}_{\bar{s}}^B$ Matrix multiplication needs at least $O(n^{2.37})$ time 

How to avoid calculating $\mathbf{Q}_{\bar{s}}^B$ and get $\mathbf{N}_{\bar{s}}\mathbf{I}$ directly and quickly?

$$\mathbf{N}_{\bar{s}}\mathbf{I} \Longleftrightarrow (\mathbf{Q}_{\bar{s}}^1\mathrm{I} + \mathbf{Q}_{\bar{s}}^2\mathrm{I} + \mathbf{Q}_{\bar{s}}^3\mathrm{I} + \ldots + \mathbf{Q}_{\bar{s}}^B\mathrm{I})$$

Matrix multiplication vector

$$\mathrm{Q}_{\bar{\mathrm{s}}}^t = \mathrm{Q}_{\bar{s}}(\mathrm{Q}_{\bar{\mathrm{s}}}^{t-1}\mathrm{I})$$

Iterative calculation

Time Complexity

Computing $\mathbf{N}_{\bar{s}}\mathbf{I}$ → B iterations → Time Complexity $O(Bm)$

Select the node with the largest marginal gain → Computing $\mathbf{N}_{\bar{s}}\mathbf{I}$ $O(n)$ time

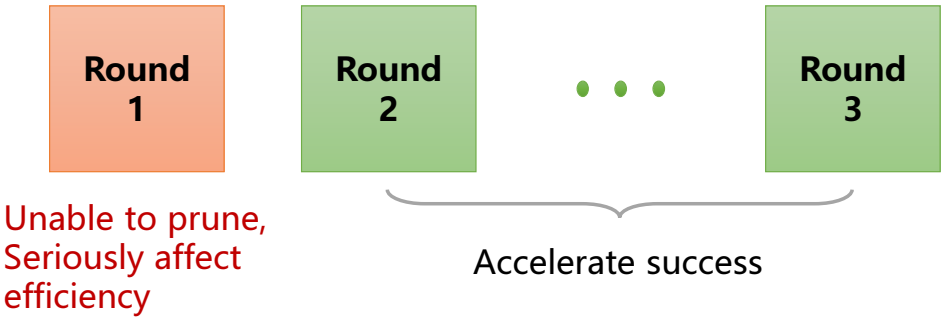
How to avoid calculating the marginal gain of all nodes?

K iterations → Time Complexity $O(kBnm)$

Solutions : BoundSel

Core idea

Submodularity ➡ Pruning, if marginal gain of round *i* ≤ upper bound of the marginal gain of round *i*-1



How to estimate the upper bound of the first-round marginal gain of nodes?

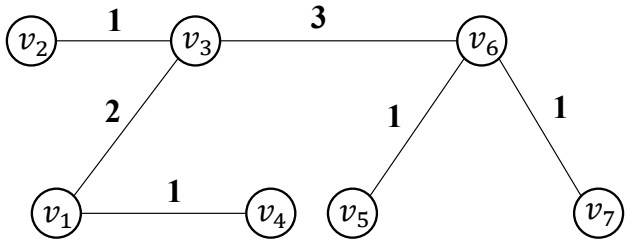
Lemma: first-round marginal gain of node v $\mathcal{G}(v,S) \leq (B-\rho)n + \rho + \sum_{t=1}^{\rho-1}(\rho - t)F_v^t$

F_v^t : the expected number of nodes that can reach v at step *t*

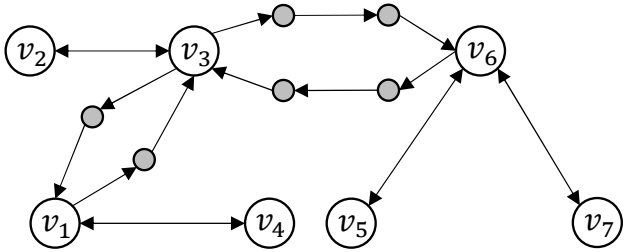
ρ : the minimal integer that satisfies $\sum_{t=1}^{\rho} F_v^t \geq n-1$

Solutions : Apply our solutions to C-WRWD

An edge-weighted graph



An unweighted graph



The expected hitting step from u to S in an unweighted graph is equal to the expected hitting cost in an edge-weighted graph.

Outline

- Problem Definition
- Applications
- Solutions
- Experiments

Experiments: Datasets

Dataset	n	m	maxD	avgD	weighted
WSN	54	2458	54	46	Yes
CELE	297	2345	134	14.5	Yes
Adolescent	2.5k	13k	27	5.2	Yes
CaGrQc	5.2K	28.9K	81	5.5	No
Advogato	6.5K	51k	804	7.8	Yes
CaHepPh	12k	237k	491	19.7	No
CondMat	16k	48k	107	5.9	Yes
Slashdot	77k	937k	2537	12.1	No
YouTube	1135k	2988k	28754	5.3	No

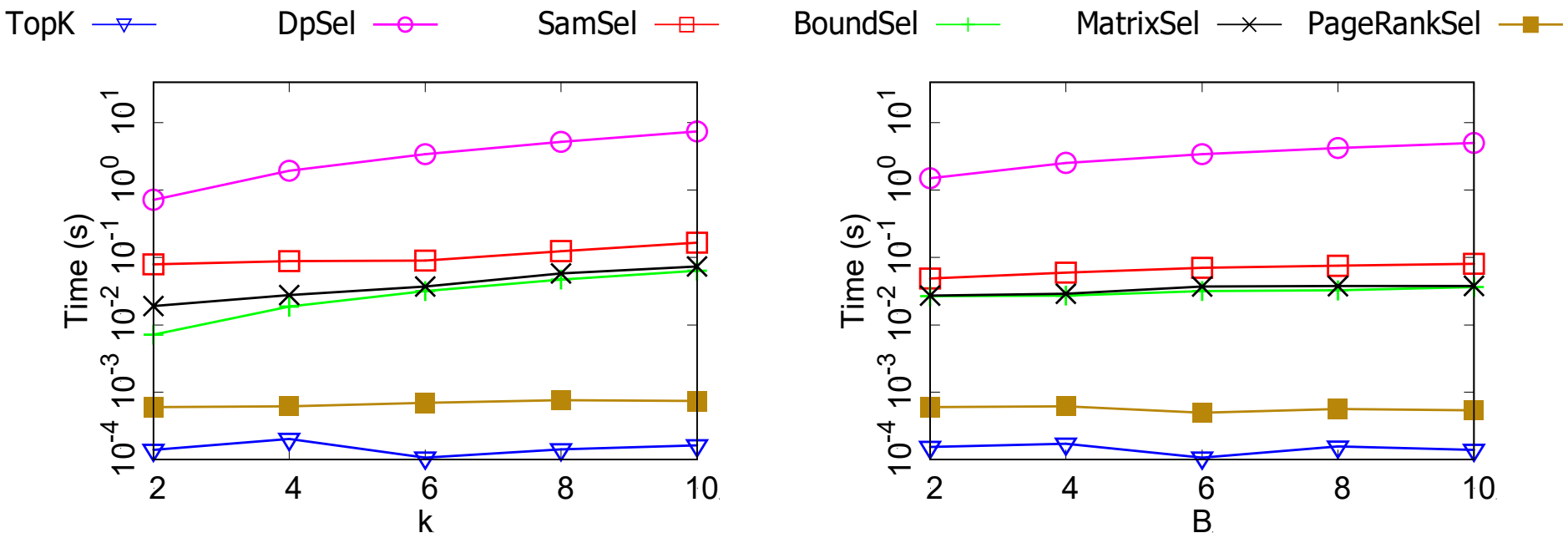
- ***n*:** node number
- ***m*:** edge number
- **maxD:** maximal degree
- **avgD:** average degree

Experiments: Algorithms

- **TopK:** selects the top-k nodes in term of degree as the seeds
- **DpSel**: a dynamic programming based greedy method
- **MatrixSel**: a matrix-based greedy method
- **BoundSel**: a bound-based greedy method
- **SamSel**: a sampling-based method proposed in [1]
- **PageRankSel**: chooses the top-k nodes in terms of the pagerank value

[1] Rong-Hua Li, et. al. 2014. Random-walk domination in large graphs. In ICDE. IEEE, 736–747.

Experiments: Efficiency

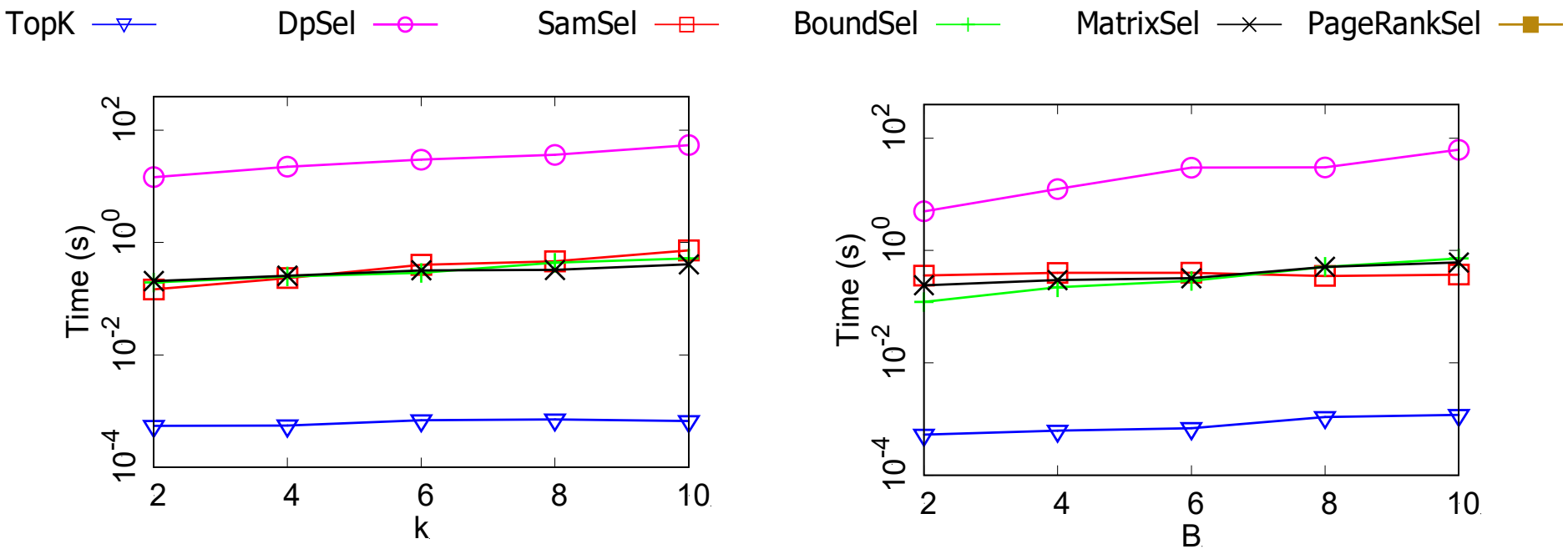


Running time vs. *k*

Running time vs. *B*

Efficiency of solutions over **PR**

Experiments: Efficiency

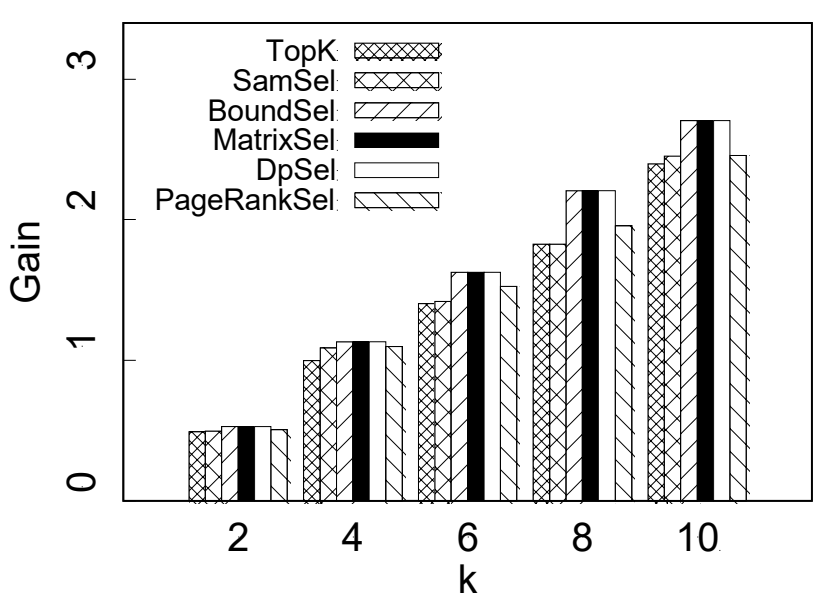


Running time vs. *k*

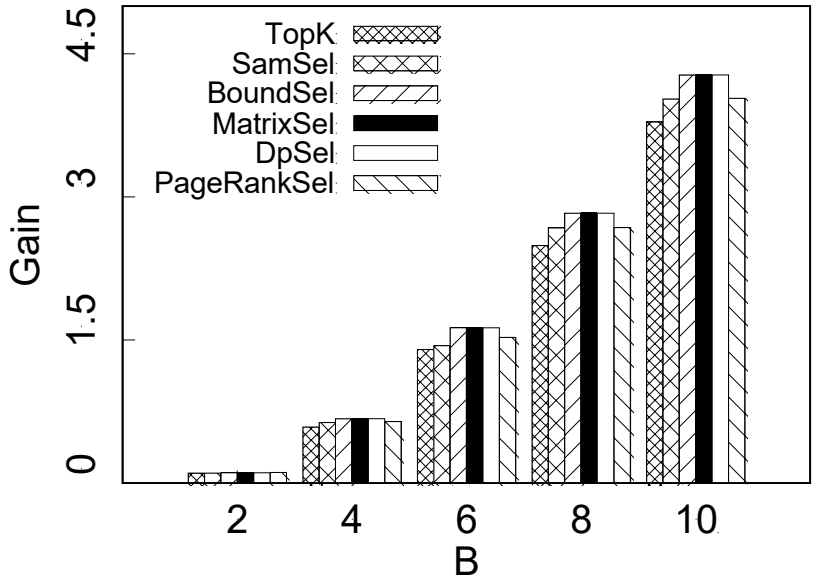
Running time vs. *B*

Efficiency of solutions over **CR**

Experiments: Effectiveness



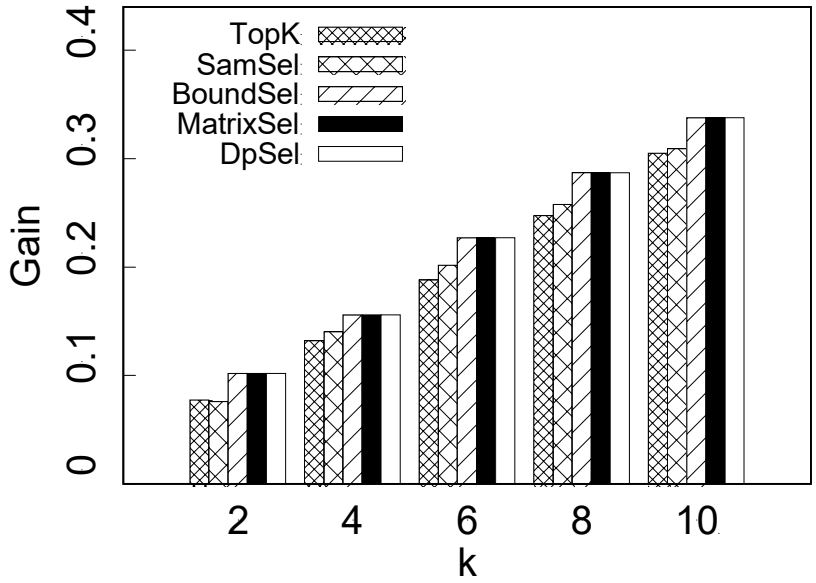
Gain vs. *k*



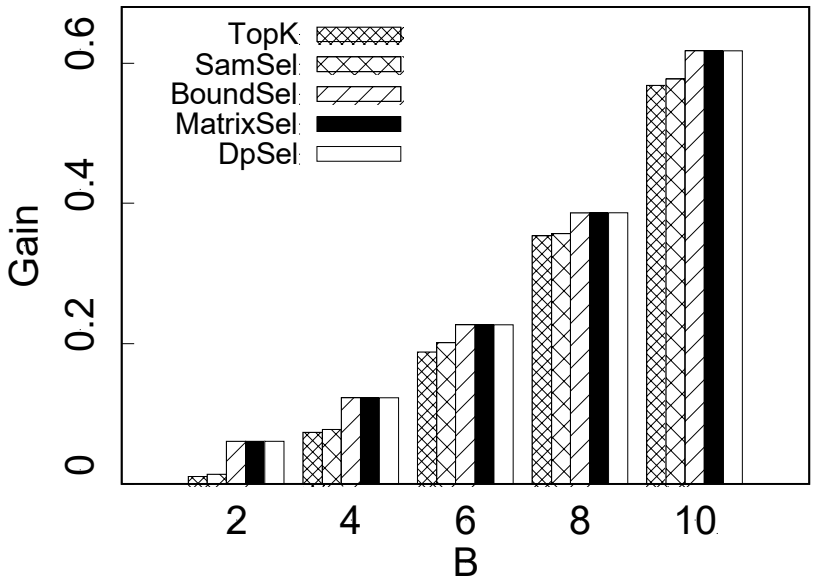
Gain vs. *B*

Effectiveness of solutions over **PR**

Experiments: Effectiveness



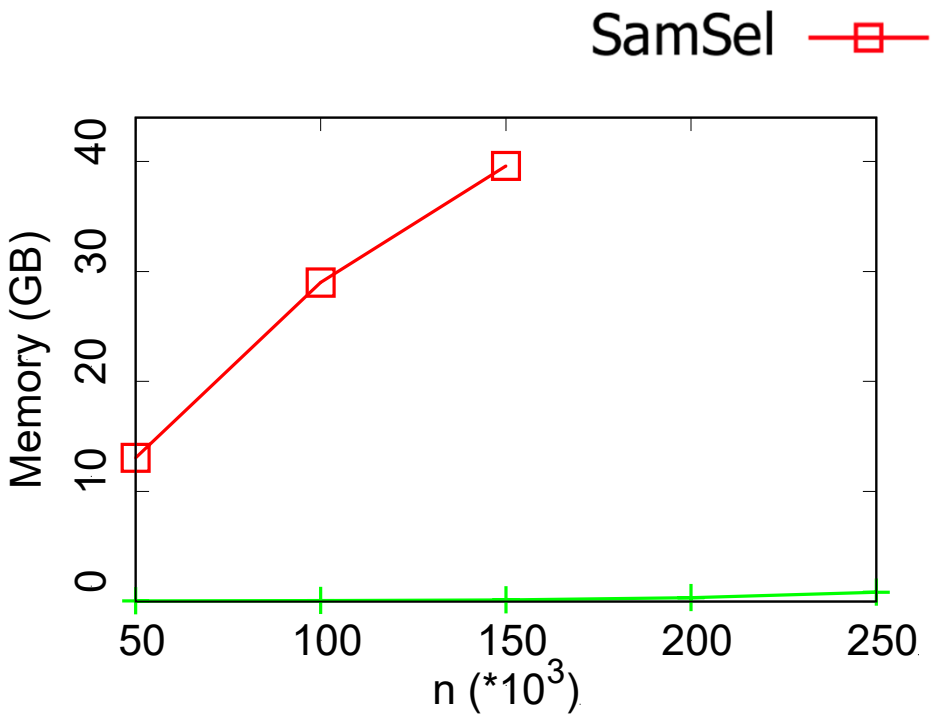
Gain vs. *k*



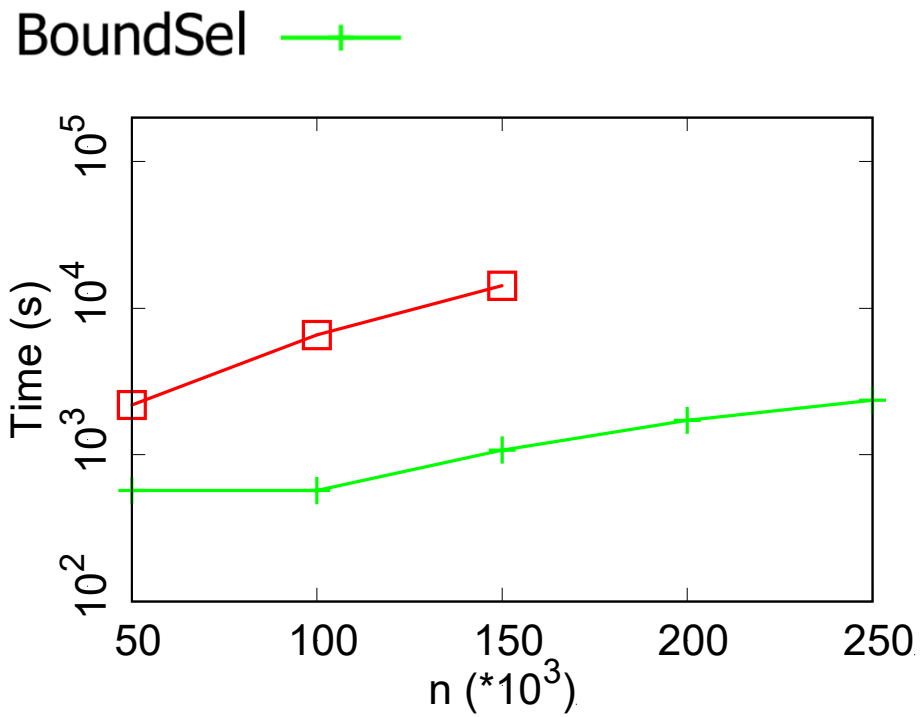
Gain vs. *B*

Effectiveness of solutions over **CR**

Experiments: Scalability



Memory vs. *n*



Running time vs. *n*

Thanks!