Matrix Scaling: A New Heuristic for the Feedback Vertex Set Problem

James Shook[1] Isabel Beichl[1]

[1]National Institute of Standards and Technology

June 10, 2014

Feedback Vertex Sets

- $G = (V, A)$ are digraphs.

Feedback Vertex Sets

- $G = (V, A)$ are digraphs.
- If G does not have a directed cycle, then it is said to be acyclic (DAG).

Feedback Vertex Sets

- $G = (V, A)$ are digraphs.
- If G does not have a directed cycle, then it is said to be acyclic (DAG).
- A set $F \subseteq V(G)$ is said to be a **feedback vertex set**, denoted by FVS, if for any cycle C in G some vertex of C is in F.

Feedback Vertex Sets

- $G = (V, A)$ are digraphs.
- If G does not have a directed cycle, then it is said to be acyclic (DAG).
- A set $F \subseteq V(G)$ is said to be a **feedback vertex set**, denoted by FVS, if for any cycle C in G some vertex of C is in F.
- An FVS is said to be **minimal** if no proper subset is an FVS.

Feedback Vertex Sets

- $G = (V, A)$ are digraphs.
- If G does not have a directed cycle, then it is said to be acyclic (DAG).
- A set $F \subseteq V(G)$ is said to be a **feedback vertex set**, denoted by FVS, if for any cycle C in G some vertex of C is in F.
- An FVS is said to be **minimal** if no proper subset is an FVS.
- We are interested in finding a minimum FVS.

Feedback Vertex Sets

- $G = (V, A)$ are digraphs.
- If G does not have a directed cycle, then it is said to be acyclic (DAG).
- A set $F \subseteq V(G)$ is said to be a **feedback vertex set**, denoted by FVS, if for any cycle C in G some vertex of C is in F.
- An FVS is said to be **minimal** if no proper subset is an FVS.
- We are interested in finding a minimum FVS.
- The order of a minimum FVS is denoted by $\tau(G)$.

Feedback Vertex Sets

- $G = (V, A)$ are digraphs.
- If G does not have a directed cycle, then it is said to be acyclic (DAG).
- A set $F \subseteq V(G)$ is said to be a **feedback vertex set**, denoted by FVS, if for any cycle C in G some vertex of C is in F.
- An FVS is said to be **minimal** if no proper subset is an FVS.
- We are interested in finding a minimum FVS.
- The order of a minimum FVS is denoted by $\tau(G)$.
- Minimizing $\tau(G)$ is NP-Hard [Karp, 1972].

Motivations

- Finding feedback vertex sets in dependency digraphs can be used to resolve deadlock.

Motivations

- Finding feedback vertex sets in dependency digraphs can be used to resolve deadlock.
- Selecting flip-flops in partial scan designs. It is a technique used in design for testing.

Three Main Steps

Most FVS heuristics follow these steps.

1. Digraph reductions: Removing vertices and arcs without changing the problem.

Three Main Steps

Most FVS heuristics follow these steps.

1. Digraph reductions: Removing vertices and arcs without changing the problem.
2. Vertex selection: Choose a vertex to be in a FVS.

Three Main Steps

Most FVS heuristics follow these steps.

1. Digraph reductions: Removing vertices and arcs without changing the problem.
2. Vertex selection: Choose a vertex to be in a FVS.
3. Removing redundant vertices: The FVS may not be minimal.

Strongly Connected Components

Definition

A digraph is said to be strongly connected if there is a directed path between any two vertices.

Strongly Connected Components

Definition

A digraph is said to be strongly connected if there is a directed path between any two vertices.

- Every arc in a strongly connected digraph is in a cycle.

Strongly Connected Components

Definition

A digraph is said to be strongly connected if there is a directed path between any two vertices.

- Every arc in a strongly connected digraph is in a cycle.
- We can use Tarjan's Algorithm [Tarjan, 1972] to reduce a digraph into strongly connected components (SCC).

Strongly Connected Components

Definition

A digraph is said to be strongly connected if there is a directed path between any two vertices.

- Every arc in a strongly connected digraph is in a cycle.
- We can use Tarjan's Algorithm [Tarjan, 1972] to reduce a digraph into strongly connected components (SCC).
- $O(|V| + |E|)$ running time

Levy and Low reductions

Definition

We call the operation of removing a vertex v from a graph G and adding the edges $N^{-}(v) \times N^{+}(v)$ that are not already in G an **exclusion** of v from G.

Levy and Low reductions

Definition

We call the operation of removing a vertex v from a graph G and adding the edges $N^-(v) \times N^+(v)$ that are not already in G an **exclusion** of v from G.

- *loop*(v): if there exists a loop, then it is in every FVS and we can safely remove it and add it to our FVS.

Levy and Low reductions

Definition

We call the operation of removing a vertex v from a graph G and adding the edges $N^{-}(v) \times N^{+}(v)$ that are not already in G an **exclusion** of v from G.

- $loop(v)$: if there exists a loop, then it is in every FVS and we can safely remove it and add it to our FVS.
- $in0\_out0(v)$: If v has no successors or predecessors, then v is not in a minimum FVS and we can safely remove it.

Levy and Low reductions

Definition

We call the operation of removing a vertex v from a graph G and adding the edges $N^{-}(v) \times N^{+}(v)$ that are not already in G an **exclusion** of v from G.

- $loop(v)$: if there exists a loop, then it is in every FVS and we can safely remove it and add it to our FVS.
- $in0\_out0(v)$: If v has no successors or predecessors, then v is not in a minimum FVS and we can safely remove it.
- $in1\_out1(v)$: If v has exactly one successor or one predecessor u, then whenever v is in a FVS so is u. Thus, we can safely exclude v from G.

Levy and Low reductions

Definition

We call the operation of removing a vertex v from a graph G and adding the edges $N^{-}(v) \times N^{+}(v)$ that are not already in G an **exclusion** of v from G.

- $loop(v)$: if there exists a loop, then it is in every FVS and we can safely remove it and add it to our FVS.
- $in0\_out0(v)$: If v has no successors or predecessors, then v is not in a minimum FVS and we can safely remove it.
- $in1\_out1(v)$: If v has exactly one successor or one predecessor u, then whenever v is in a FVS so is u. Thus, we can safely exclude v from G.
- The operations can be done in any order [Levy and Low, 1988].

fvs_Max_Deg

Choosing a vertex based off of vertex degrees is quicker.

Algorithm 1: MaxDeg

Data: A Digraph $G = (X, U)$
Result: A FVS S

```
begin
    S ⟵ ∅
    LL_graph_reductions(G, S)
    L ⟵ get_SCC(G)
    while |L| ≠ 0 do
        remove g from L
        v ⟵ max(min(d⁺(v), d⁻(v))|v ∈ V(G))
        remove v from g
        S ⟵ S + {v}
        LL_reductions(g, S)
        L ⟵ get_SCC(g) + L
    end
    S ⟵ remove_redundant_nodes(G, S)
    return S
end
```

Mean return time

- The probability that a vertex x of a cycle C is in a minimum FVS is at least $\frac{1}{|C|}$.

Mean return time

- The probability that a vertex x of a cycle C is in a minimum FVS is at least $\frac{1}{|C|}$.
- It is reasonable to suspect that a vertex that is in a lot of small cycles is in a minimum FVS.

Mean return time

- The probability that a vertex x of a cycle C is in a minimum FVS is at least $\frac{1}{|C|}$.
- It is reasonable to suspect that a vertex that is in a lot of small cycles is in a minimum FVS.
- Speckenmeyer [1990] and Lemaic and Speckenmeyer [2009] studied a random walks on a digraph and calculated the stationary distribution of the transition matrix.

Mean return time

- The probability that a vertex x of a cycle C is in a minimum FVS is at least $\frac{1}{|C|}$.
- It is reasonable to suspect that a vertex that is in a lot of small cycles is in a minimum FVS.
- Speckenmeyer [1990] and Lemaic and Speckenmeyer [2009] studied a random walks on a digraph and calculated the stationary distribution of the transition matrix.
- They selected the vertex with the smallest mean return time.

Mean return time

- The probability that a vertex x of a cycle C is in a minimum FVS is at least $\frac{1}{|C|}$.
- It is reasonable to suspect that a vertex that is in a lot of small cycles is in a minimum FVS.
- Speckenmeyer [1990] and Lemaic and Speckenmeyer [2009] studied a random walks on a digraph and calculated the stationary distribution of the transition matrix.
- They selected the vertex with the smallest mean return time.
- Their method operates in about $O(|F|n^{2.376})$ time.

MFVSmean

Algorithm 2: MFVSmean

Data: A Digraph $G = (X, U)$
Result: A FVS S
begin
$S \longleftarrow \emptyset$
$LL\_graph\_reductions(G, S)$
$L \longleftarrow get\_SCC(G)$
while $|L| \neq 0$ **do**
remove g from L
$v \longleftarrow MFVSmean\_selection(g)$
remove v from g
$S \longleftarrow S + \{v\}$
$LL\_reductions(g, S)$
$L \longleftarrow get\_SCC(g) + L$
end
$S \longleftarrow remove\_redundant\_nodes(G, S)$
return S
end

MFVSmean

Algorithm 3: MFVSmean_selection

Data: A Digraph $G = (X, U)$
Result: A vertex v
begin
  $\mathbf{P} \longleftarrow \mathit{CreateTransitionMatrix}(G)$
  $\pi' \longleftarrow \mathit{ComputeStationaryDistributionVector}(\mathbf{P})$
  $\mathbf{P} \longleftarrow \mathit{CreateTransitionMatrix}(G^{-1})$
  $\pi'' \longleftarrow \mathit{ComputeStationaryDistributionVector}(\mathbf{P})$
  $\pi \longleftarrow \pi' + \pi''$
  determine $v \in V$ with $\pi_v = \|\pi\|_\infty$
  return v
end

Disjoint Cycle Unions and FVS

A set of vertex disjoint cycles is said to be a disjoint cycles union (DCU).

Disjoint Cycle Unions and FVS

A set of vertex disjoint cycles is said to be a disjoint cycles union (DCU).
If S is an FVS, then there exists an $x \in S$ such that it is in at least $\frac{|DCU(G)|}{|S|}$ DCUs.

Disjoint Cycle Unions and FVS

A set of vertex disjoint cycles is said to be a disjoint cycles union (DCU).

If S is an FVS, then there exists an $x \in S$ such that it is in at least $\frac{|DCU(G)|}{|S|}$ DCUs.

- DCUs are not a local property.

Disjoint Cycle Unions and FVS

A set of vertex disjoint cycles is said to be a disjoint cycles union (DCU).
If S is an FVS, then there exists an $x \in S$ such that it is in at least $\frac{|DCU(G)|}{|S|}$ DCUs.

- DCUs are not a local property.
- It is reasonable to suspect that a vertex that is in many DCUs is in a minimum FVS.

Disjoint Cycle Unions and FVS

A set of vertex disjoint cycles is said to be a disjoint cycles union (DCU).
If S is an FVS, then there exists an $x \in S$ such that it is in at least $\frac{|DCU(G)|}{|S|}$ DCUs.

- DCUs are not a local property.
- It is reasonable to suspect that a vertex that is in many DCUs is in a minimum FVS.
- Finding all DCUs is hard.

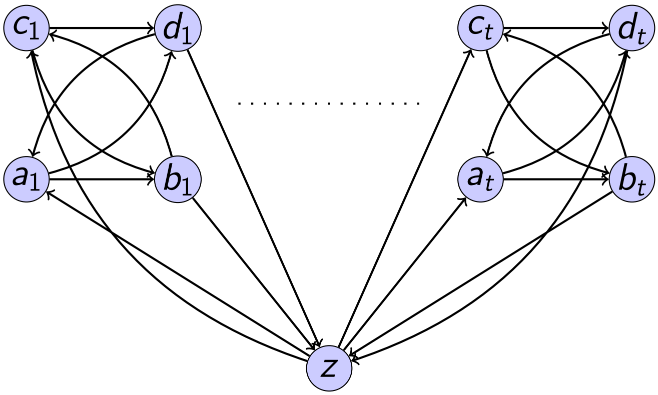


Figure: For $t \geq 2$ the vertex z is not in a minimum FVS, but is in nearly every DCU and most cycles.

Disjoint Cycle Unions and the Permanent

The permanent of a matrix $\mathbf{A}$ is defined as

$$perm(\mathbf{A}) = \sum_{\sigma} \prod_{i=1}^{n} a_{i,\sigma(i)}.$$

Disjoint Cycle Unions and the Permanent

The permanent of a matrix **A** is defined as

$$perm(\mathbf{A}) = \sum_{\sigma} \prod_{i=1}^{n} a_{i,\sigma(i)}.$$

The permanent counts the number of spanning disjoint cycle unions. We can create an auxilary digraph H from G by adding loops to the vertices of G.

Disjoint Cycle Unions and the Permanent

The permanent of a matrix $\mathbf{A}$ is defined as

$$perm(\mathbf{A}) = \sum_{\sigma} \prod_{i=1}^{n} a_{i,\sigma(i)}.$$

The permanent counts the number of spanning disjoint cycle unions. We can create an auxilary digraph H from G by adding loops to the vertices of G.

$$perm(\mathbf{A}(H)) - 1 = |DCU(G)|$$

m_balance

Let H be the auxilary digraph created as before. From H we can create a matrix called the m_balance($\mathbf{A}$(H)) that gives the fraction of DCUs that every arc of H is in.

$$m\_bal(\mathbf{A}(H)) = \frac{a_{i,j} \times perm(\mathbf{A}(H)_{i,j})}{perm(\mathbf{A}(H))}. \tag{1}$$

m_balance

Let H be the auxilary digraph created as before. From H we can create a matrix called the m_balance($\mathbf{A}$(H)) that gives the fraction of DCUs that every arc of H is in.

$$m\_bal(\mathbf{A}(H)) = \frac{a_{i,j} \times perm(\mathbf{A}(H)_{i,j})}{perm(\mathbf{A}(H))}. \tag{1}$$

- The m_balance is doubly stochastic since every vertex is incident with every DCU.

m_balance

Let H be the auxilary digraph created as before. From H we can create a matrix called the m_balance($\mathbf{A}$(H)) that gives the fraction of DCUs that every arc of H is in.

$$m\_bal(\mathbf{A}(H)) = \frac{a_{i,j} \times perm(\mathbf{A}(H)_{i,j})}{perm(\mathbf{A}(H))}. \qquad (1)$$

- The m_balance is doubly stochastic since every vertex is incident with every DCU.
- The loop with the smallest value in the m_balance corresponds to the vertex that is in the most DCUs.

*m*_balance

Let H be the auxilary digraph created as before. From H we can create a matrix called the m_balance($\mathbf{A}$(H)) that gives the fraction of DCUs that every arc of H is in.

$$m\_bal(\mathbf{A}(H)) = \frac{a_{i,j} \times perm(\mathbf{A}(H)_{i,j})}{perm(\mathbf{A}(H))}. \qquad (1)$$

- The m_balance is doubly stochastic since every vertex is incident with every DCU.
- The loop with the smallest value in the m_balance corresponds to the vertex that is in the most DCUs.
- The m_balance is very hard to calculate.

Sinkhorn Balancing.

Algorithm 4: Sinkhorn_Selection

```
Data: A Digraph G = (V, U)
Result: A vertex v
begin
    A ⟵ adjacency matrix of G
    A ⟵ add ones to the diagonal of A
    for i ∈ {1, ..., ⌈log(n)⌉} do
        A ⟵ normalize the rows of A
        A ⟵ normalize the columns of A
    end
    v is the vertex corresponding to the lowest value on the
    diagonal of A
    return v
end
```

- Soules [1991] showed that Algorithm 4 converges quickly if $\mathbf{A}$ is totally supported. Reducing to strongly connected components guarantees this.

Sinkhorn Balancing.

Algorithm 5: Sinkhorn_Selection

Data: A Digraph $G = (V, U)$
Result: A vertex v
begin
  $\mathbf{A} \longleftarrow$ adjacency matrix of G
  $\mathbf{A} \longleftarrow$ add ones to the diagonal of $\mathbf{A}$
  for $i \in \{1, \ldots, \lceil log(n) \rceil\}$ **do**
    $\mathbf{A} \longleftarrow$ normalize the rows of $\mathbf{A}$
    $\mathbf{A} \longleftarrow$ normalize the columns of $\mathbf{A}$
  end
  v is the vertex corresponding to the lowest value on the diagonal of $\mathbf{A}$
  return v
end

- Soules [1991] showed that Algorithm 4 converges quickly if **A** is totally supported. Reducing to strongly connected components guarantees this.
- Beichl and Sullivan [1999] showed that the limiting matrix of the Sinkhorn-Knopp algorithm can be used to estimate the permanent of **A**.

Sinkhorn Balancing.

Algorithm 6: Sinkhorn_Selection

Data: A Digraph $G = (V, U)$
Result: A vertex v
begin
  $\mathbf{A} \longleftarrow$ adjacency matrix of G
  $\mathbf{A} \longleftarrow$ add ones to the diagonal of $\mathbf{A}$
  for $i \in \{1, ..., \lceil log(n) \rceil\}$ **do**
    $\mathbf{A} \longleftarrow$ normalize the rows of $\mathbf{A}$
    $\mathbf{A} \longleftarrow$ normalize the columns of $\mathbf{A}$
  end
  v is the vertex corresponding to the lowest value on the diagonal of $\mathbf{A}$
  return v
end

- Soules [1991] showed that Algorithm 4 converges quickly if $\mathbf{A}$ is totally supported. Reducing to strongly connected components guarantees this.
- Beichl and Sullivan [1999] showed that the limiting matrix of the Sinkhorn-Knopp algorithm can be used to estimate the permanent of $\mathbf{A}$.
- We observed that we only need to complete $log(n)$ iterations for the order to settle down.

fvs_sh_del

Algorithm 7: FVS_SH_Del

Data: A Digraph $G = (X, U)$
Result: A FVS S
begin
  $H \longleftarrow G$
  $S \longleftarrow \emptyset$
  $LL\_graph\_reductions(H, S)$
  $L \longleftarrow get\_SCC(H)$
  while $|L| \neq 0$ **do**
    remove g from L
    $v \longleftarrow Sinkhorn\_selection(g)$
    remove v from g
    $S \longleftarrow S + \{v\}$
    $LL\_reductions(g, S)$
    $L \longleftarrow get\_SCC(g) + L$
  end
  $S \longleftarrow remove\_redundant\_nodes(G, S)$
  return S
end

$$O(|S| log(n) n^2)$$

fvs_sh_del_mod

Algorithm 8: FVS_SH_DEL_MOD

Data: A Digraph $G = (X, U)$
Result: A FVS S
begin
  $H \longleftarrow G$
  $S \longleftarrow \emptyset$
  $LL\_graph\_reductions(H, S)$
  while $|V(H)| \neq 0$ **do**
    $v \longleftarrow Sinkhorn\_selection(H)$
    remove v from H
    $S \longleftarrow S + \{v\}$
    $LL\_reductions(H, S)$
  end
  $S \longleftarrow remove\_redundant\_nodes(G, S)$
  return S
end

Remove Redundant Vertices

1. Let $S = S_0$ and assume S_0 is in the reverse order in that the vertices of S where selected by Algorithm 7.

Remove Redundant Vertices

1. Let $S = S_0$ and assume S_0 is in the reverse order in that the vertices of S where selected by Algorithm 7.
2. We then recursively select vertex v_i from S_{i-1} and check to see if $G - (S_{i-1} - \{v\})$ is a DAG.

Remove Redundant Vertices

1. Let $S = S_0$ and assume S_0 is in the reverse order in that the vertices of S where selected by Algorithm 7.
2. We then recursively select vertex v_i from S_{i-1} and check to see if $G - (S_{i-1} - \{v\})$ is a DAG.
3. If it is not a DAG, then we let $S_i = S_{i-1}$.

Remove Redundant Vertices

1. Let $S = S_0$ and assume S_0 is in the reverse order in that the vertices of S where selected by Algorithm 7.
2. We then recursively select vertex v_i from S_{i-1} and check to see if $G - (S_{i-1} - \{v\})$ is a DAG.
3. If it is not a DAG, then we let $S_i = S_{i-1}$.
4. If it is a DAG, then v is redundant and we let $S_i = S_{i-1} - \{v\}$.

Lower Bounds

- An FVS S is said to be an ϵ-approximation if $|S| \leq \epsilon\tau(G)$.

Lower Bounds

- An FVS S is said to be an ϵ-approximation if $|S| \leq \epsilon\tau(G)$.
- If $t \leq \tau(G)$, then S is an $\frac{|S|}{t}$-approximation.

Lower Bounds

- An FVS S is said to be an ϵ-approximation if $|S| \leq \epsilon\tau(G)$.
- If $t \leq \tau(G)$, then S is an $\frac{|S|}{t}$-approximation.
- Let X be a set of cycles and $c_X(x)$ be the number of cycles that hit x.

Lower Bounds

- An FVS S is said to be an ϵ-approximation if $|S| \leq \epsilon\tau(G)$.
- If $t \leq \tau(G)$, then S is an $\frac{|S|}{t}$-approximation.
- Let X be a set of cycles and $c_X(x)$ be the number of cycles that hit x.
- If T is an FVS, then

$$\sum_{v \in T} c_X(v) \geq |X|. \tag{2}$$

Lower Bounds

- An FVS S is said to be an ϵ-approximation if $|S| \leq \epsilon\tau(G)$.
- If $t \leq \tau(G)$, then S is an $\frac{|S|}{t}$-approximation.
- Let X be a set of cycles and $c_X(x)$ be the number of cycles that hit x.
- If T is an FVS, then

$$\sum_{v \in T} c_X(v) \geq |X|. \tag{2}$$

- Let α, β be orderings of $V(G)$ and T respectively such that $c_X(\alpha_i) \geq c_X(\alpha_{i+1})$ and β and $c_X(\beta_i) \geq c_X(\beta_{i+1})$

Lower Bounds

- An FVS S is said to be an ϵ-approximation if $|S| \leq \epsilon\tau(G)$.
- If $t \leq \tau(G)$, then S is an $\frac{|S|}{t}$-approximation.
- Let X be a set of cycles and $c_X(x)$ be the number of cycles that hit x.
- If T is an FVS, then

$$\sum_{v \in T} c_X(v) \geq |X|. \tag{2}$$

- Let α, β be orderings of $V(G)$ and T respectively such that $c_X(\alpha_i) \geq c_X(\alpha_{i+1})$ and β and $c_X(\beta_i) \geq c_X(\beta_{i+1})$
- $$\sum_{i=0}^{t} c_X(\alpha_i) \geq |X| \qquad \sum_{i=0}^{t'} c_X(\beta_i) \geq |X|.$$

Lower Bounds

- An FVS S is said to be an ϵ-approximation if $|S| \leq \epsilon\tau(G)$.
- If $t \leq \tau(G)$, then S is an $\frac{|S|}{t}$-approximation.
- Let X be a set of cycles and $c_X(x)$ be the number of cycles that hit x.
- If T is an FVS, then

$$\sum_{v \in T} c_X(v) \geq |X|. \tag{2}$$

- Let α, β be orderings of $V(G)$ and T respectively such that $c_X(\alpha_i) \geq c_X(\alpha_{i+1})$ and β and $c_X(\beta_i) \geq c_X(\beta_{i+1})$
- $$\sum_{i=0}^{t} c_X(\alpha_i) \geq |X| \qquad \sum_{i=0}^{t'} c_X(\beta_i) \geq |X|.$$
- $$k\tau(G) \geq \sum_{i=0}^{t} c_X(\alpha_i) \geq |X| = \epsilon|S|.$$

Random digraphs

- We created Erdos-Renyi random digraphs by visiting every ordered pair of vertices and placing an arc with probability p between them.

Random digraphs

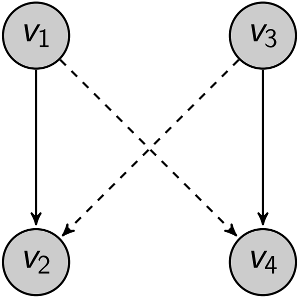
- We created Erdos-Renyi random digraphs by visiting every ordered pair of vertices and placing an arc with probability p between them.
- A digraph is k-regular if $d^+(v) = d^-(v) = k$ for every $v \in V(G)$.

Random digraphs

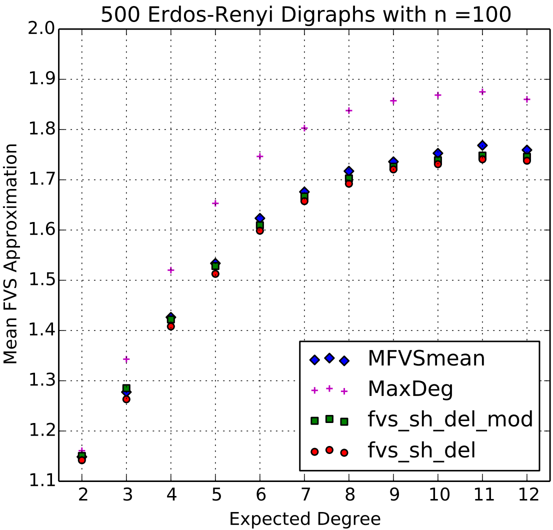
- We created Erdos-Renyi random digraphs by visiting every ordered pair of vertices and placing an arc with probability p between them.
- A digraph is k-regular if $d^+(v) = d^-(v) = k$ for every $v \in V(G)$.
- We chose random k-regular digraphs uniformly by first using the methods of Kleitman-Wang to create a k-regular digraph.

Random digraphs

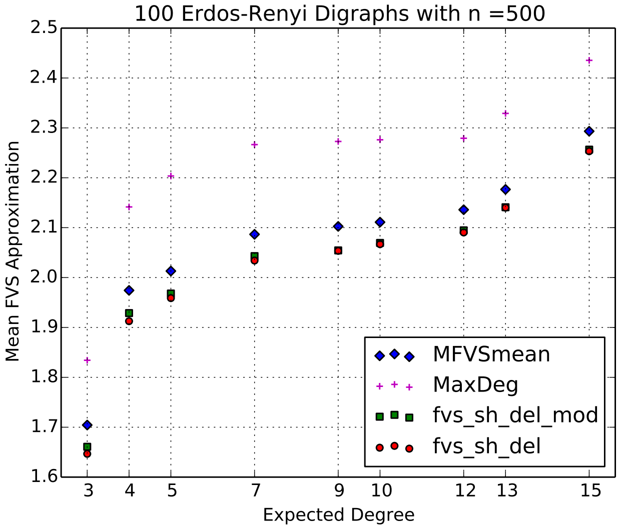
- We created Erdos-Renyi random digraphs by visiting every ordered pair of vertices and placing an arc with probability p between them.
- A digraph is k-regular if $d^+(v) = d^-(v) = k$ for every $v \in V(G)$.
- We chose random k-regular digraphs uniformly by first using the methods of Kleitman-Wang to create a k-regular digraph.
- We then perform $k^2 n$ arc switches to simulate a uniformly chosen one.


v_1
v_3
v_2
v_4

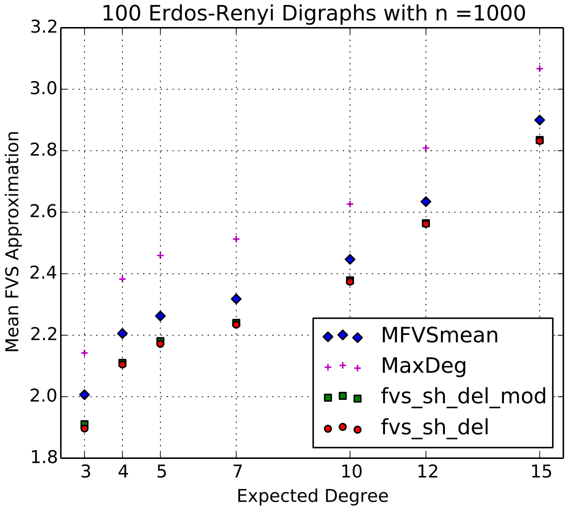
Erdos-Renyi Random Digraphs n = 100



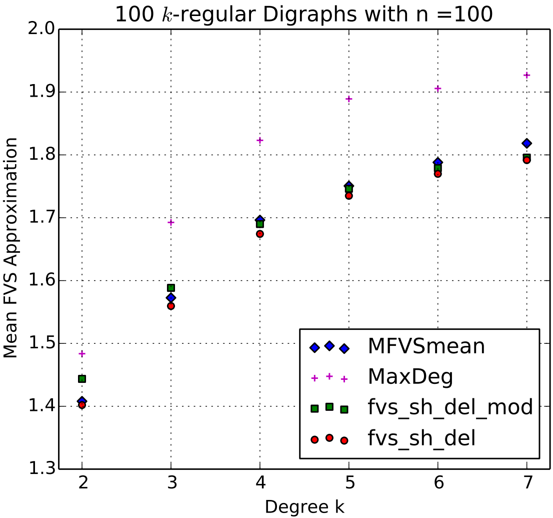
Erdos-Renyi Random Digraphs n = 500



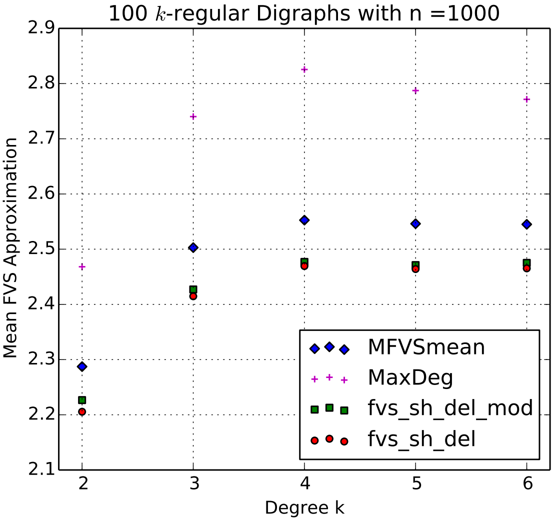
Erdos-Renyi Random Digraphs n = 1000



k-Regular Digraphs n = 100



k-Regular Digraphs n = 1000



Entropy

For many small digraphs the Sinkhorn method performed better than the m_balance.

Entropy

For many small digraphs the Sinkhorn method performed better than the *m*_balance. The entropy of a doubly stochastic matrix **A** is

$$entropy(\mathbf{A}) = -\sum_{i,j} a_{i,j} log(a_{i,j}).$$

Entropy

For many small digraphs the Sinkhorn method performed better than the m_balance. The entropy of a doubly stochastic matrix $\mathbf{A}$ is

$$entropy(\mathbf{A}) = -\sum_{i,j} a_{i,j} log(a_{i,j}).$$

Beichl and Sullivan showed the limiting matrix of the Sinkhorn-Knopp algorithm maximizes the entropy for all doubly-stochastic matrices with a given zero-one pattern.

Isabel Beichl and Francis Sullivan. Approximating the permanent via importance sampling with application to the dimer covering problem. *Journal of Computational Physics*, 149(1):128 – 147, 1999. ISSN 0021-9991. doi: http://dx.doi.org/10.1006/jcph.1998.6149. URL http://www.sciencedirect.com/science/article/pii/S0021999198961496.

R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

Mile Lemaic and Ewald Speckenmeyer. Markov-chain-based heuristics for the minimum feedback vertex set problem. Technical report, 2009. URL http://e-archive.informatik.uni-koeln.de/596/.

Hanoch Levy and David W Low. A contraction algorithm for finding small cycle cutsets. *Journal of Algorithms*, 9(4):470 – 493, 1988. ISSN 0196-6774. doi: http://dx.doi.org/10.1016/0196-6774(88)90013-2. URL http://www.sciencedirect.com/science/article/pii/0196677488900132.

George W. Soules. The rate of convergence of Sinkhorn balancing. *Linear Algebra and its Applications*, 150:3–40, May 1991. ISSN 00243795. doi: 10.1016/0024-3795(91)90157-R. URL http://www.sciencedirect.com/science/article/pii/002437959190157Rhttp://linkinghub.elsevier.com/retrieve/pii/002437959190157R.

Ewald Speckenmeyer. On feedback problems in digraphs. In Manfred Nagl, editor, *Graph-Theoretic Concepts in Computer Science*, volume 411 of *Lecture Notes in Computer Science*, pages 218–231. Springer Berlin Heidelberg, 1990. ISBN 978-3-540-52292-8. doi: 10.1007/3-540-52292-1_16. URL http://dx.doi.org/10.1007/3-540-52292-1_16.

Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.