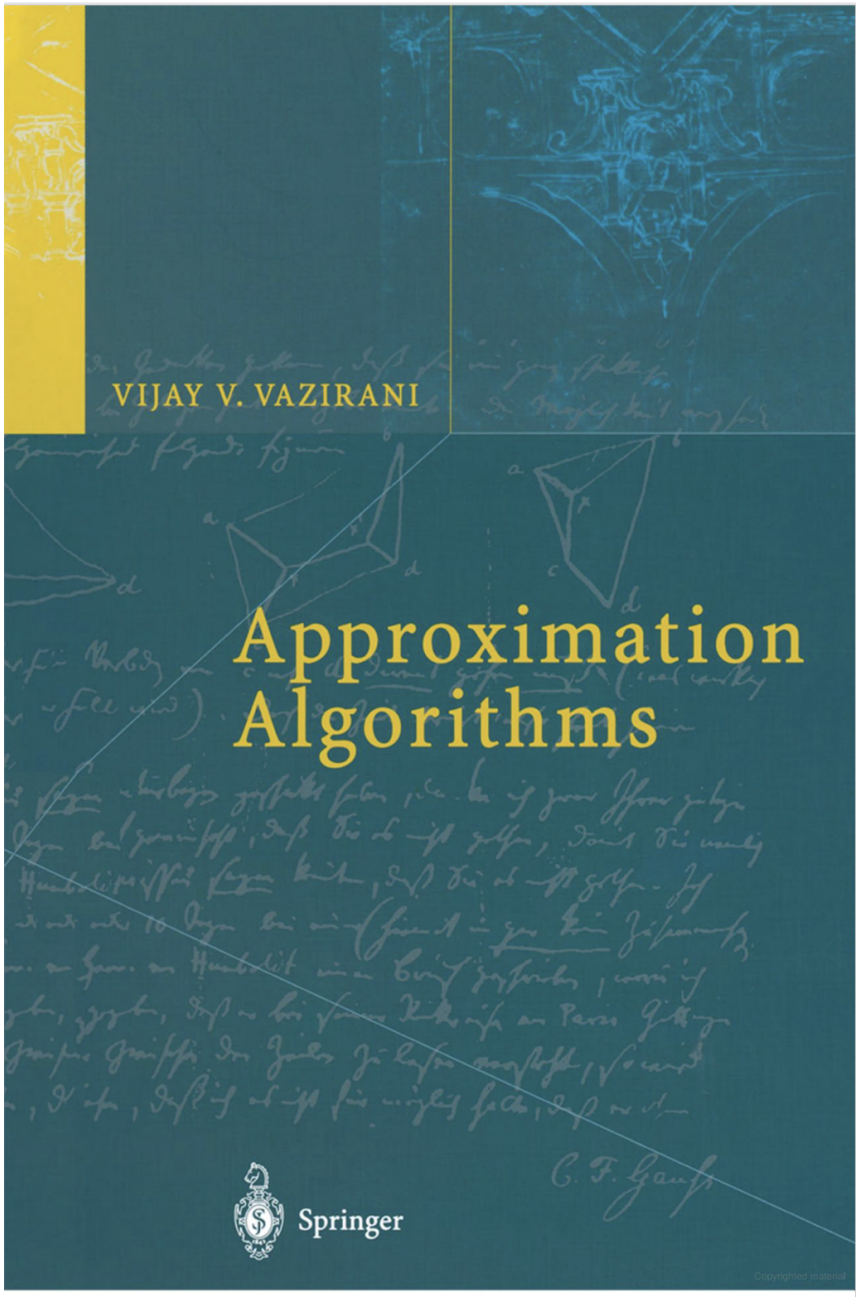
Approximation Algorithms for the Matching Augmentation Problem

by Joseph Cheriyan, C&O Dept., U.Waterloo
(based on joint work with R.Cummings, J.Dippel, J.Zhu)

Approximation algorithms



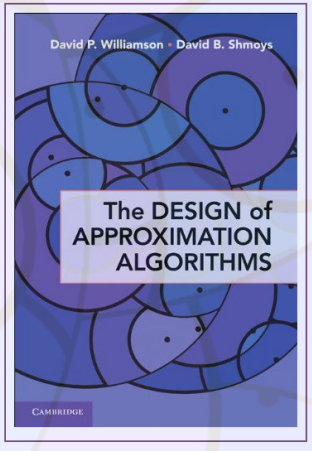
Approximation algorithms

The Design of Approximation Algorithms

by David P. Williamson and David B. Shmoys

- Home
- Download
- Order
- FAQs
- Errata
- Contact

This is the companion website for the book *The Design of Approximation Algorithms* by David P. Williamson and David B. Shmoys, published by Cambridge University Press.



Interesting discrete optimization problems are everywhere, from traditional operations research planning problems, such as scheduling, facility location, and network design, to computer science problems in databases, to advertising issues in viral marketing. Yet most interesting discrete optimization problems are NP-hard. Thus unless P = NP, there are no efficient algorithms to find optimal solutions to such problems. This book shows how to design *approximation algorithms*: efficient algorithms that find provably near-optimal solutions.

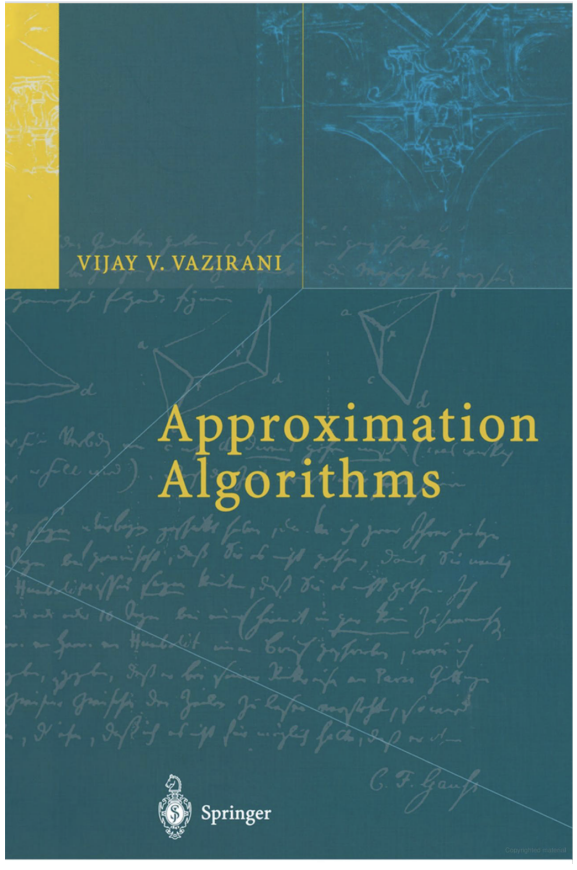
The book is organized around several central algorithmic techniques for designing approximation algorithms, including greedy and local search algorithms, dynamic programming, linear and semidefinite programming, and randomization. Each chapter in the first part of the book is devoted to a single algorithmic technique, which is then applied to several different problems. The second part revisits the techniques, but offers more sophisticated treatments of them. The book also covers methods for proving that optimization problems are hard to approximate.

Designed as a textbook for graduate-level algorithms courses, the book will also serve as a reference for researchers who are interested in the heuristic solution of discrete optimization problems.

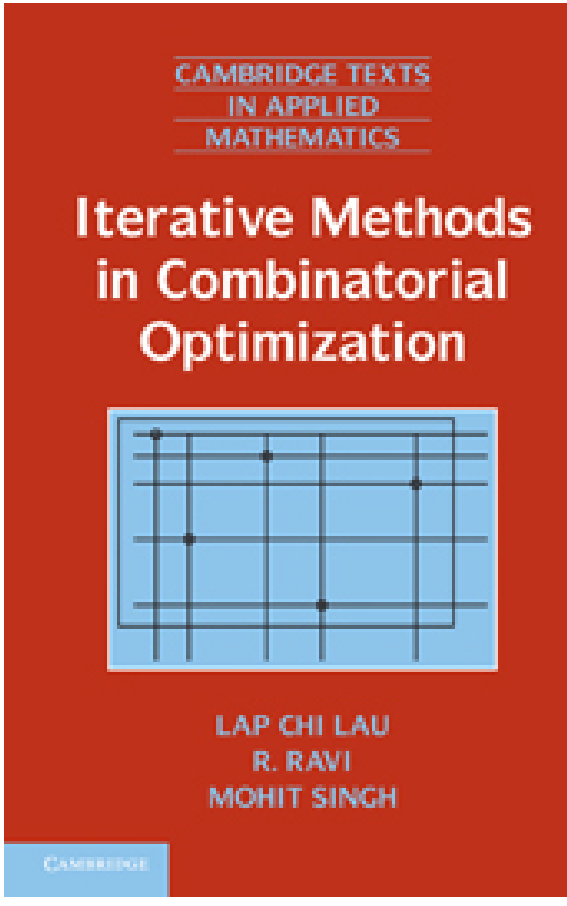
An electronic-only edition of the book is provided in our Download section.

This website by DnA Design, Copyright 2010.

Approximation algorithms and Combinatorial optimization



Copyright © 2010 by David P. Williamson and David B. Shmoys. All rights reserved. To be published by Cambridge University Press.



Minimum cost network design (. . . in real life)

Given n towns and "connection-costs" between each pair, find a cheapest subgraph that connects all pairs of towns.

MST (minimum spanning tree) problem

Given n towns and "connection-costs" between each pair, find a cheapest subgraph that connects all pairs of towns even after the failure of one link.

2-Node connectivity

Graph (undirected) $G = (V, E)$ with $|V| \geq 3$ is 2-node connected:

$\iff$ $G - v$ is connected for all $v \in V$

$\iff$ each neighborhood $N_G(S)$ has ≥ 2 nodes, where $1 \leq |S| \leq |V| - 2$

$\iff$ G has 2 openly-disjoint (internally node-disjoint) v, w-paths, $\forall v, w \in V (v \neq w)$ (Menger's theorem)

2-Edge connectivity

Graph (undirected) $G = (V, E)$ is 2-edge connected:

$\iff$ $G - e$ is connected for all $e \in E$

$\iff$ each cut $\delta(S) = (S, \bar{S})$ has ≥ 2 edges, where $\emptyset \neq S \subsetneq V$

$\iff$ G has 2 edge-disjoint v, w-paths, $\forall v, w \in V (v \neq w)$
(Menger's theorem)

Min-cost 2-ECSS

Given an (undirected) graph $G = (V, E)$, and edge-costs $c : E \to \mathbf{R}_+$, design an algorithm to find a 2-edge connected spanning subgraph (2-ECSS) of minimum cost.

This problem is NP-hard. (A polynomial-time algorithm for finding an optimal solution would imply **P**=**NP**.)

Revised goal:
α-approximation algorithm ($\alpha \in \mathbf{R}_+$):
Design & analyze an algorithm to find a 2-ECSS $G^{alg} = (V, F)$ s.t. cost$(G^{alg}) = c(F) \leq \alpha \, \mathrm{OPT}(G)$.

Approximation algorithms for min-cost 2-ECSS (quick look)

Restrictions on edge costs	Approx.Ratio	Authors & Journal
$c \in \mathbf{R}_+$	2	Khuller & Vishkin, JACM (1994)
$c \in \mathbf{R}_+$	2	Jain, CCA (2001) Iterative rounding
unit costs	4/3	Sebo & Vygen, CCA (2014)
zero/one costs		
Forest Augmentation (FAP)	2	(above)
Tree Augmentation (TAP)	3/2	Kortsarz & Nutov, TALG (2016)
Matching Augmentation (MAP)	7/4	C., Dippel, Grandoni, Khan, Narayan Math.Prog. (2020)

An Improved Approximation Algorithm for the Matching Augmentation Problem

`https://arxiv.org/abs/2007.11559`

Abstract: We present a $\frac{5}{3}$-approximation algorithm for the matching augmentation problem (MAP): ...

arXiv.org > cs > arXiv:2007.11559

Search...

Help | Advar

Computer Science > Data Structures and Algorithms

[Submitted on 22 Jul 2020]

An Improved Approximation Algorithm for the Matching Augmentation Problem

J.Cheriyan, R.Cummings, J.Dippel, J.Zhu

We present a $\frac{5}{3}$-approximation algorithm for the matching augmentation problem (MAP): given a multi-graph with edges of cost either zero or one such that the edges of cost zero form a matching, find a 2-edge connected spanning subgraph (2-ECSS) of minimum cost.
A $\frac{7}{4}$-approximation algorithm for the same problem was presented recently, see Cheriyan, et al., "The matching augmentation problem: a $\frac{7}{4}$-approximation algorithm," {\em Math. Program.}, 182(1):315--354, 2020; arXiv:1810.07816.
Our improvement is based on new algorithmic techniques, and some of these may lead to advances on related problems.

Comments: 23 pages
Subjects: **Data Structures and Algorithms (cs.DS)**; Discrete Mathematics (cs.DM)
MSC classes: 68W25, 90C59, 90C27, 68R10, 05C85
Cite as: arXiv:2007.11559 [cs.DS]
(or arXiv:2007.11559v1 [cs.DS] for this version)

Warm-up: 2-approximation for unit-cost 2-ECSS

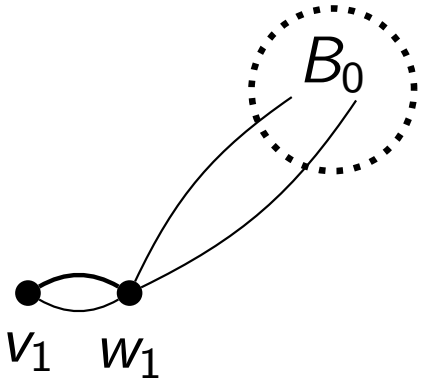
Design & analyze an algorithm to find a 2-ECSS $G^{alg} = (V, F)$ s.t. $\text{cost}(G^{alg}) = c(F) = |F| \leq 2 \cdot \text{OPT}(G)$.

Key points:

- Lower-bound on $\text{OPT}(G)$: $n := |V(G)|$.
- Ear decomposition of $G = (V, E)$: a partition of E into paths or cycles, $P_0, P_1, \ldots, P_k$, such that P_0 is the trivial path with one node, and each P_i $(1 \leq i \leq k)$ is either
  (1) a path that has both end nodes in $V_{i-1} = V(P_0) \cup V(P_1) \cup \ldots \cup V(P_{i-1})$ but has no internal nodes in V_{i-1}, or
  (2) a cycle that has exactly one node in V_{i-1}.

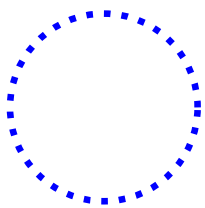
Theorem: Graph G is 2EC (2-edge connected) iff G has an ear-decomposition.

Ear-decomposition of G, where G has cut nodes (not 2NC)

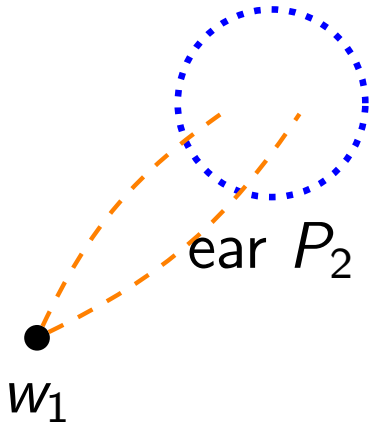


2EC graph G.

$B_0 = P_0 \cup P_1$



$B_0 = P_0 \cup P_1$



$B_0 = P_0 \cup P_1$

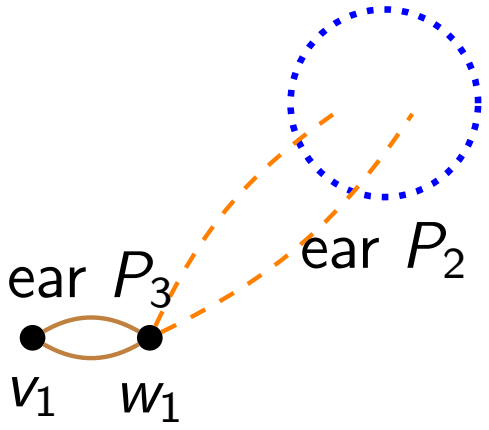


Figure: Ear decomposition of a 2EC graph G. Start with ear decomposition $P_0 \cup P_1$ of B_0.

Warm-up: 2-approximation for unit-cost 2-ECSS

Design & analyze an algorithm to find a 2-ECSS $G^{alg} = (V, F)$ s.t. $\text{cost}(G^{alg}) = c(F) = |F| \leq 2 \cdot \text{OPT}(G)$.
Key points:

- Lower-bound on $\text{OPT}(G)$: $n := |V(G)|$.
- Ear decomposition of $G = (V, E)$: a partition of E into paths or cycles, $P_0, P_1, \ldots, P_k$, such that ...

Credit scheme: Assign \$2 to each node.
(Total credit $= 2\cdot$ (Lower bound).)
Algorithm & analysis: Construct ear decomposition of G, but discard ears of length$=1$.
For each ear P_i of length≥ 2, buy edges of P_i using credit on internal nodes of P_i.

Next: 2-approximation for MAP (zero/one edge costs)

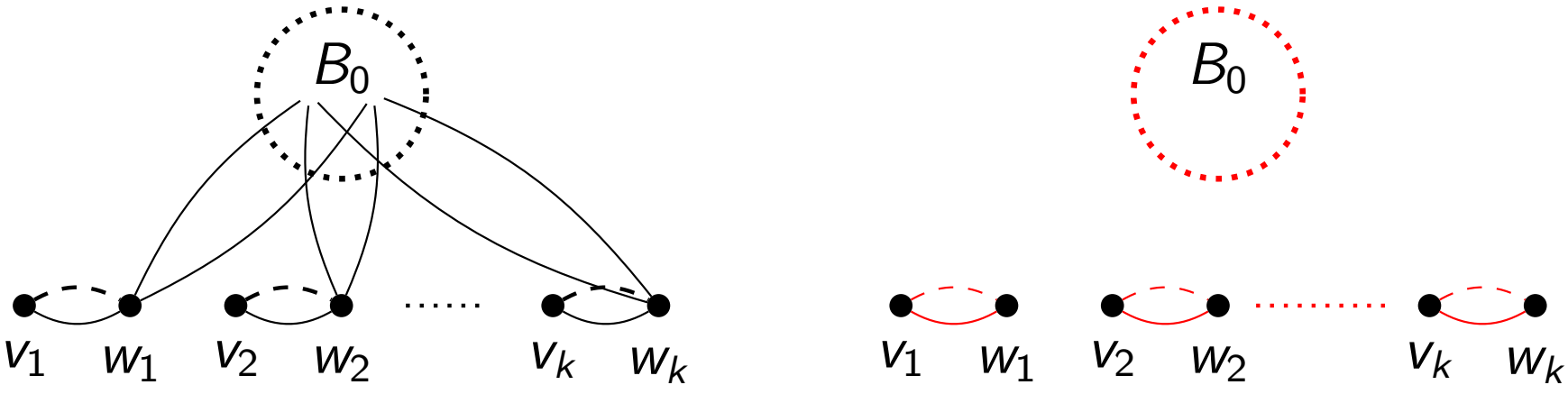
Given $G = (V, E)$ and matching $M \subset E$;
edge-costs: $c(e) = 0$ if $e \in M$, and $c(e) = 1$ otherwise.

Design & analyze an algorithm to find a 2-ECSS $G^{alg} = (V, F)$
s.t. cost$(G^{alg}) = c(F) \leq 2 \cdot \mathrm{OPT}(G)$.

Lower-bound on $\mathrm{OPT}(G)$: min cost of 2-edge cover of G.
2-edge cover of G: subgraph that has degree ≥ 2 at each node.
$D2(G)$: a subgraph of min cost that has degree ≥ 2 at each node.
- $\mathrm{OPT}(G) \geq c(D2(G))$.

Credit scheme: Assign \$2 to each unit-edge of $D2(G)$.
(Total credit $= 2\cdot$ (Lower bound).)

$\mathrm{OPT}(G)$ versus $D2(G)$: G has cut nodes (not 2NC)



$$\mathrm{OPT}(G) = c(B_0) + 3k. \qquad c(D2(G)) = c(B_0) + k.$$

Figure: Edges of cost zero and one are illustrated by dashed and solid lines, respectively. For $k \gg c(B_0)$, $\frac{\mathrm{OPT}(G)}{c(D2(G))} \approx 3$. Cut-nodes are an "obstruction" for proving approximation factor 2.

2-approximation for MAP (zero/one edge costs)

Assume sub-instance G_i is 2NC and simple.
Lower-bound on $\mathrm{OPT}(G_i)$:

$$\mathrm{OPT}(G_i) \geq c(D2(G_i)).$$

Credit scheme: Assign \$2 to each unit-edge of $D2(G_i)$.
(Total credit $= 2\cdot$ (Lower bound).)

Each unit-edge of $D2(G_i)$ pays \$1, and this suffices to buy all edges of $D2(G_i)$.

Each unit-edge of $D2(G_i)$ keeps \$1 credit.

Algorithm :

- Preprocess: if G has cut nodes, decompose G into sub-instances $G_1, G_2, \ldots, G_k$ corresponding to blocks of G. Each sub-instance is 2NC (hence, simple≡no parallel edges); solve each sub-instance (separately).
- Return the union of the 2ECSSs $H_1, H_2, \ldots, H_k$ of $G_1, G_2, \ldots, G_k$.

2-approximation for MAP (zero/one edge costs)

Assume sub-instance G_i is 2NC and simple.
Lower-bound on $\mathrm{OPT}(G_i)$: $\mathrm{OPT}(G_i) \geq c(D2(G_i))$, where $D2(G_i)$ is a min-cost subgraph of G_i with minimum degree 2.
Credit scheme: Each unit-edge of $D2(G_i)$ keeps \$1 credit.

Algorithm :

- Preprocess: if G has cut nodes, decompose G into sub-instances $G_1, G_2, \ldots, G_k$ corresponding to blocks of G. Each sub-instance is 2NC (hence, simple≡no parallel edges); solve each sub-instance (separately).
  - compute $H_i := D2(G_i)$;
  - apply Bridge-Covering to H_i: augment edges to each connected component of H_i to make it 2EC;
  - apply Gluing to H_i: augment edges to merge the connected components of H_i to obtain a 2ECSS of G_i;
- Return the union of the 2ECSSs $H_1, H_2, \ldots, H_k$ of $G_1, G_2, \ldots, G_k$.

Gluing step on current graph H

apply Gluing to H: augment edges to merge the connected components of H to obtain a 2ECSS of G;

assume each zero-edge is in $D2(G)$, so each zero-edge is in H;

Credit scheme: Assume that each connected component of H has credit of $\geq$ \$2.

Contract each connected component of H to obtain $\tilde{H}$.

Apply algorithm for unit-cost 2-ECSS.

Recall: For each ear P_i of length≥ 2, buy edges of P_i using credit on internal nodes of P_i.

Bridge-covering step on current graph H

apply bridge-covering to a connected component C_0 of H: augment edges to C_0 obtain a 2EC connected component of H;

Bridge-covering step on current graph H

apply bridge-covering to a connected component C_0 of H: augment edges to C_0 obtain a 2EC connected component of H;

Credit scheme: Re-assign credit of \$1 on each unit-edge of H:

- each connected component of H has \$1 c-credit;
- each 2EC-block (of any connected component) of H has \$1 b-credit;
- each bridge (of any connected component) of H has \$1 credit.

Bridge-covering step on current graph H: Informally

apply bridge-covering to a connected component C_0 of H: augment edges to C_0 obtain a 2EC connected component of H;

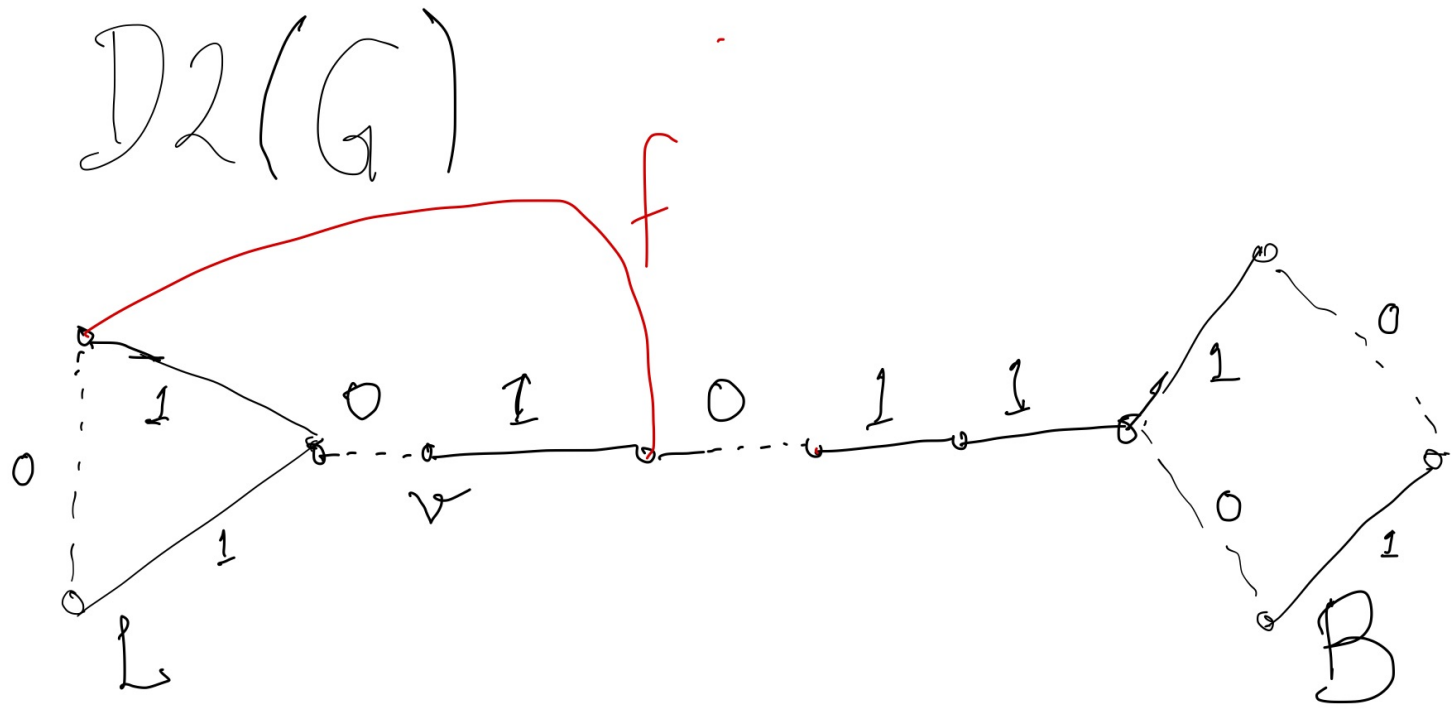
See figure on next slide.

Let C_0 be a connected component of $H := D2(G)$ that has a sequence of bridges between two 2EC-blocks L (on left) and B (on right).

Assume: all inter-component edges are incident to B, so any edge incident to $C_0 - V(B)$ has both ends in C_0.

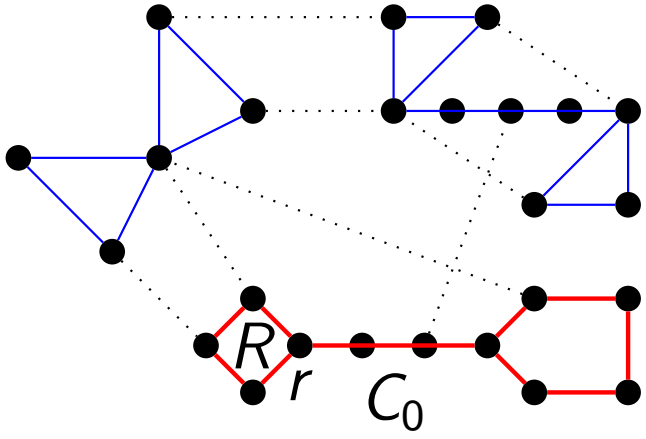
Algorithm: pick an edge $f \in E(G) - E(H)$ with one end in L and the other end "farthest" from L.

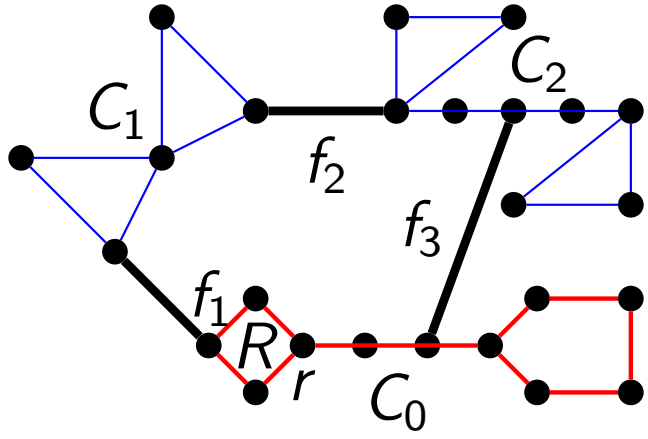
- If f covers a unit-bridge e of C_0, use \$1 credit of e to buy f.
- Otherwise, neighbour v of L in C_0 is a cut node of G. Contradiction to: G is 2NC.



- component has 1 c-credit
- each 2EC-block has 1 b-credit
- each unit-bridge has 1 credit

Bridge covering via pseudo-ears





A *pseudo-ear* of H w.r.t. C_0 starting at R is a sequence $R, f_1, C_1, f_2, C_2, \ldots, f_{k-1}, C_{k-1}, f_k$, where $C_0, C_1, \ldots, C_{k-1}$ are distinct connected components of H, $f_1, \ldots, f_k \in E(G) - E(H)$, each f_i, $i \in [k-1]$, has one end node in C_{i-1} and the other end node in C_i, f_1 has an end node in R, and f_k has one end node in C_{k-1} and one end node in $C_0 - V(R)$. The end node of f_k in $C_0 - V(R)$ is called the *head node* of the pseudo-ear.

An Improved Approximation Algorithm for the Matching Augmentation Problem

`https://arxiv.org/abs/2007.11559`

Abstract: We present a $\frac{5}{3}$-approximation algorithm for the matching augmentation problem (MAP): ...

arXiv.org > cs > arXiv:2007.11559

Search...

Help | Advar

Computer Science > Data Structures and Algorithms

[Submitted on 22 Jul 2020]

An Improved Approximation Algorithm for the Matching Augmentation Problem

J.Cheriyan, R.Cummings, J.Dippel, J.Zhu

We present a $\frac{5}{3}$-approximation algorithm for the matching augmentation problem (MAP): given a multi-graph with edges of cost either zero or one such that the edges of cost zero form a matching, find a 2-edge connected spanning subgraph (2-ECSS) of minimum cost.
A $\frac{7}{4}$-approximation algorithm for the same problem was presented recently, see Cheriyan, et al., "The matching augmentation problem: a $\frac{7}{4}$-approximation algorithm," {\em Math. Program.}, 182(1):315--354, 2020; arXiv:1810.07816.
Our improvement is based on new algorithmic techniques, and some of these may lead to advances on related problems.

Comments: 23 pages
Subjects: **Data Structures and Algorithms (cs.DS)**; Discrete Mathematics (cs.DM)
MSC classes: 68W25, 90C59, 90C27, 68R10, 05C85
Cite as: arXiv:2007.11559 [cs.DS]
(or arXiv:2007.11559v1 [cs.DS] for this version)

5/3-approximation for MAP (zero/one edge costs)

Given $G = (V, E)$ and matching $M \subset E$;
edge-costs: $c(e) = 0$ if $e \in M$, and $c(e) = 1$ otherwise.

Design & analyze an algorithm to find a 2-ECSS $G^{alg} = (V, F)$
s.t. cost$(G^{alg}) = c(F) \leq \frac{5}{3} \cdot \mathrm{OPT}(G)$.

Assume sub-instance G_i is a "well-structured" instance of MAP
(defined in a following slide).
Lower-bound on $\mathrm{OPT}(G_i)$:

$$\mathrm{OPT}(G_i) \geq c(D2(G_i)).$$

Credit scheme: Assign \$$\frac{5}{3}$ to each unit-edge of $D2(G_i)$.
(Total credit $= \frac{5}{3} \cdot$ (Lower bound).)

Preprocessing for 5/3-approximation for MAP

Definition: An instance of MAP$\star$ is an instance of MAP with ≥ 12 nodes that contains

- no cut nodes,
- no parallel edges,
- no "zero split",
- no "unit split",
- no "split cycle",
- no R4 gadget, and
- no R8 gadget.

A MAP-instance G with $|V(G)| \geq 12$ can be efficiently (poly-time) decomposed into a collection of MAP-instances $G_1, \ldots, G_k$ s.t.

(a) either $|V(G_i)| < 12$ or G_i is an instance of MAP$\star$, $\forall i \in [k]$,

(b) the edge sets $E(G_1), \ldots, E(G_k)$ are pairwise disjoint, and

(c) a 2-ECSS H of G can be obtained by computing 2-ECSSes $H_1, \ldots, H_k$ of $G_1, \ldots, G_k$.

(d) Moreover, the approximation guarantee is preserved, meaning that $c(H) \leq \frac{5}{3}\mathrm{OPT}(G) - 2$ provided $c(H_i) \leq \max(\mathrm{OPT}(G_i), \frac{5}{3}\mathrm{OPT}(G_i) - 2), \forall i \in [k]$.

Preprocessing for 5/3-approximation for MAP

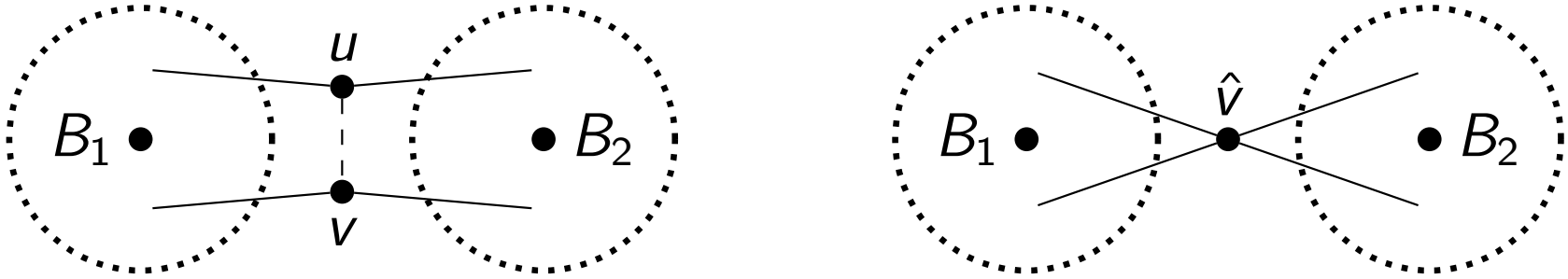


Figure: Illustration of a "zero split" uv, and its contraction. The contracted node $\hat{v}$ is a cut node.

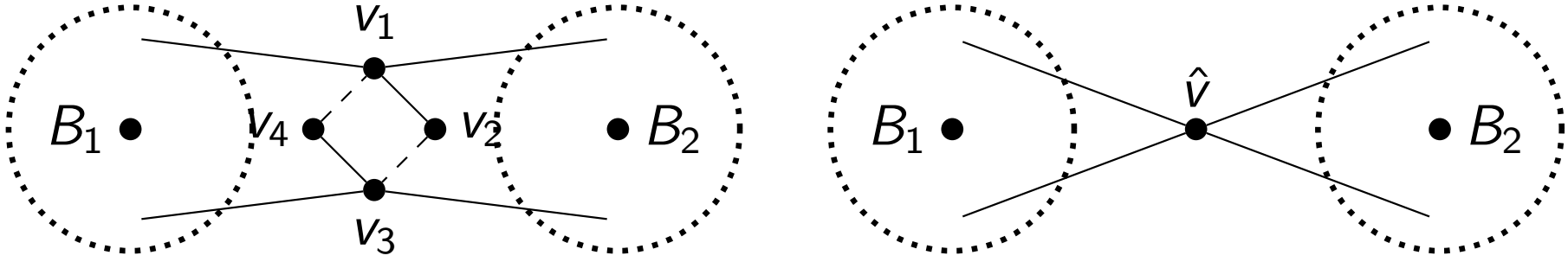
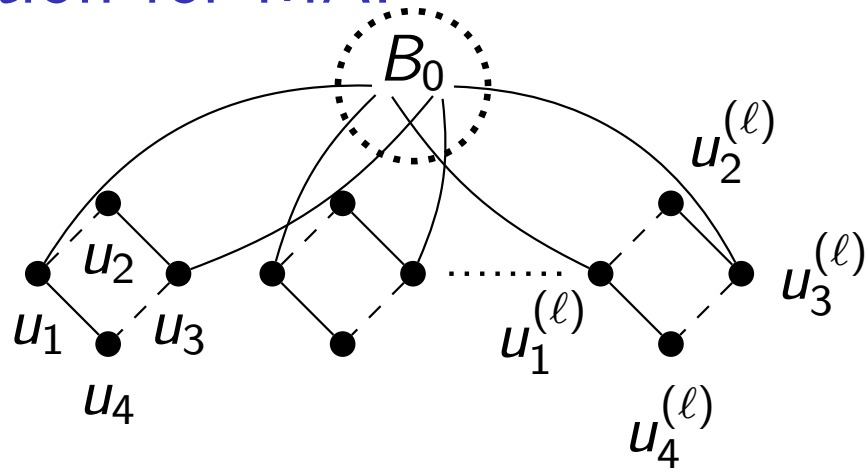
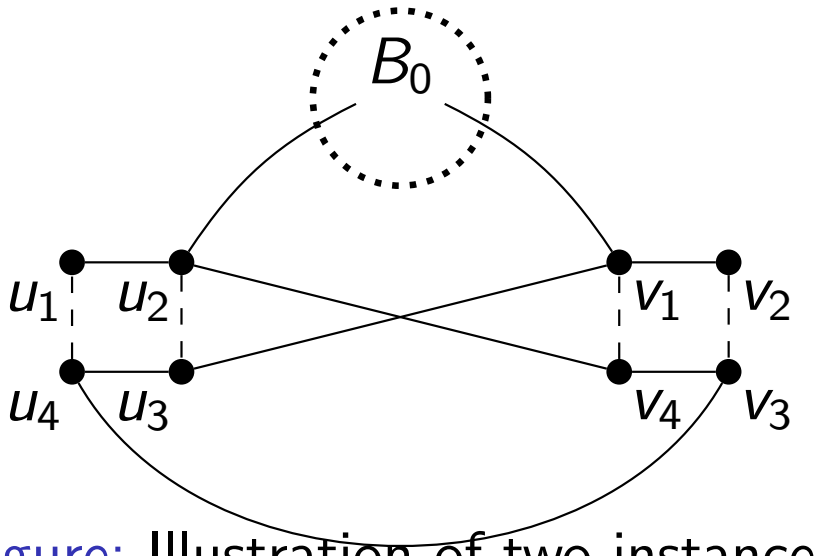


Figure: Illustration of a "split cycle", and its contraction. The subgraph C induced by $\{v_1, v_2, v_3, v_4\}$ is the "split cycle". The contracted node $\hat{v}$ is a cut node.

Preprocessing for 5/3-approximation for MAP

R4: is a four-cycle C of cost 2 such that two of the nonadjacent nodes of C have degree two in G. Several instances of **R4** are shown on the right.





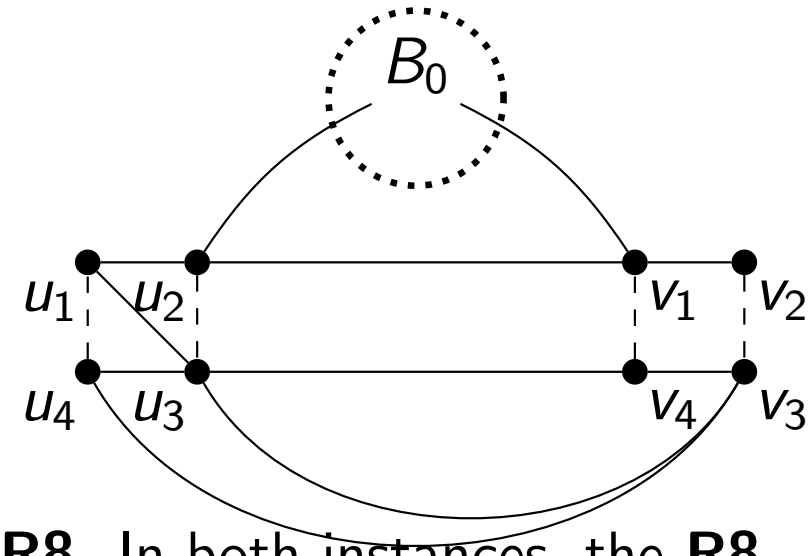


Figure: Illustration of two instances of **R8**. In both instances, the **R8** is the subgraph $\hat{C}$ induced by $\{u_1, u_2, u_3, u_4, v_1, v_2, v_3, v_4\}$; $\hat{C}$ contains two 4-cycles $C_1 = u_1, u_2, u_3, u_4, u_1$ and $C_2 = v_1, v_2, v_3, v_4, v_1$; $\hat{C}$ has exactly two attachments u_2 and v_1.

5/3-approximation for MAP (zero/one edge costs)

Assume sub-instance G_i is an instance of MAP$\star$.
Lower-bound on $\mathrm{OPT}(G_i)$: $\mathrm{OPT}(G_i) \geq c(D2(G_i))$.
Credit scheme: Each unit-edge of $D2(G_i)$ keeps $\$\frac{2}{3}$ credit.

Algorithm :

- Preprocess: decompose G into "well-structured sub-instances" (of MAP$\star$) $G_1, G_2, \ldots, G_k$.
  Each sub-instance satisfies the requirements of MAP$\star$; solve each sub-instance (separately).
  - compute $H_i := D2(G_i)$;
  - apply Bridge-Covering to H_i: augment edges to each connected component of H_i to make it 2EC;
  - apply Gluing to H_i: augment edges to merge the connected components of H_i to obtain a 2ECSS of G_i;
  - Output a 2-ECSS H of G from the 2ECSSs $H_1, H_2, \ldots, H_k$ of $G_1, G_2, \ldots, G_k$ by undoing the "preprocessing transformations".

Bridge covering via pseudo-ears

Please look up the paper on arXiv and read Section 5.2 (Analysis of a pseudo-ear augmentation), less than 3 pages!

Previous slides gave an informal explanation of the bridge covering step using credits of \$1 per unit-edge of *D2*.

More work is needed to to extend the bridge covering step to the setting of $\$\frac{2}{3}$ credits per unit-edge of *D2*, but it takes up less than 3 pages.

Gluing step on current graph H

apply Gluing to H: augment edges to merge the connected components of H (that have no bridges) to obtain a 2ECSS of G;

assume each zero-edge is in $D2(G)$, so each zero-edge is in H;

Credit scheme:
A connected component of H is called **small** if it has 2 unit-edges, and is called **large** if it has ≥ 3 unit-edges.
Assume that each large connected component has credit of $\geq \$2$, and each small connected component has credits of $\$\frac{4}{3}$.

Merge small connected components with other connected components while preserving the credit invariant, until there are no small connected components.

Contract each connected component of H to obtain $\tilde{H}$.

Apply algorithm for unit-cost 2-ECSS.

Recall: For each ear P_i of length≥ 2, buy edges of P_i using credit on internal nodes of P_i.