Randomized Contractions for Multiobjective Minimum Cuts

Hassene Aissi[1], Ali Ridha Mahjoub[2], and R. Ravi*[3]

1 **Univ. Paris-Dauphine, PSL Research University, CNRS, UMR 7243, LAMSADE, Paris, France**
aissi@lamsade.dauphine.fr

2 **Univ. Paris-Dauphine, PSL Research University, CNRS, UMR 7243, LAMSADE, Paris, France**
mahjoub@lamsade.dauphine.fr

3 **Carnegie Mellon University, Pittsburgh, USA**
ravi@andrew.cmu.edu

Abstract

We show that Karger's randomized contraction method [7] can be adapted to multiobjective global minimum cut problems with a constant number of edge or node budget constraints to give efficient algorithms.

For global minimum cuts with a single edge-budget constraint, our extension of the randomized contraction method has running time $\tilde{O}(n^3)$ in an n-node graph improving upon the best-known randomized algorithm with running time $\tilde{O}(n^4)$ due to Armon and Zwick [1]. Our analysis also gives a new upper bound of $O(n^3)$ for the number of optimal solutions for a single edge-budget min cut problem. For the case of $(k-1)$ edge-budget constraints, the extension of our algorithm saves a logarithmic factor from the best-known randomized running time of $O(n^{2k}\log^3 n)$. A main feature of our algorithms is to adaptively choose, at each step, the appropriate cost function used in the random selection of edges to be contracted.

For the global min cut problem with a constant number of node budgets, we give a randomized algorithm with running time $\tilde{O}(n^2)$, improving the current best determinisitic running time of $O(n^3)$ due to Goemans and Soto [5]. Our method also shows that the total number of distinct optimal solutions is bounded by $\binom{n}{2}$ as in the case of global min-cuts. Our algorithm extends to the node-budget constrained global min cut problem excluding a given sink with the same running time and bound on number of optimal solutions, again improving upon the best-known running time by a factor of $O(n)$. For node-budget constrained problems, our improvements arise from incorporating the idea of merging any infeasible super-nodes that arise during the random contraction process.

In contrast to cuts excluding a sink, we note that the node-cardinality constrained min-cut problem containing a given source is strongly NP-hard using a reduction from graph bisection.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases minimum cut, multiobjective optimization, budget constraints, graph algorithms, randomized algorithms

Digital Object Identifier 10.4230/LIPIcs.ESA.2017.6

* This material is based upon research supported in part by the U.S. Office of Naval Research under award number N00014-12-1-1001, the U.S. National Science Foundation under award number CCF-1527032, and a visiting professorship at LAMSADE, Paris Dauphine University.

1 Introduction

Cut problems play a central role in combinatorial optimization and arise routinely in many practical areas such as telecommunications, project networks and databases [7] as well as the bottleneck computation in the separation routine for important network optimization problems such as the TSP [12]. Let $G = (V, E)$ be an undirected simple graph with n nodes and m edges, and $c^1, \ldots, c^k : E \to \mathbb{Z}^+$ ($w^1, \ldots, w^{k-1} : V \to \mathbb{Z}^+$) be k ($k-1$) non-negative cost functions defined on the set of edges (nodes), where k is a constant. A cut X in G is a subset of nodes $X \subseteq V$ such that $\emptyset \neq X \neq V$, and it determines the set $\delta(X)$ of edges with exactly one end in X. The cost of cut X in criterion j is $c^j(\delta(X)) := \sum_{e \in \delta(X)} c^j(e)$ ($w^j(X) := \sum_{v \in X} w^j(v)$). Given $k-1$ cost bounds $b_1, \ldots, b_{k-1}$, we study the following multiobjective versions of the minimum cut problem.

- Edge-budget constraints: find a cut C^* minimizing edges cost c^k subject to the constraints $c^i(\delta(C^*)) \leq b_i$ for $i = 1, \ldots, k-1$.
- Node-budget constraints: find a cut C^* minimizing edges cost c^k subject to the constraints $w^i(C^*) \leq b_i$ for $i = 1, \ldots, k-1$.
- Node-budget constraints including a source s (excluding a sink t): given a specific node $s \in V$ ($t \in V$), find a cut C^* minimizing edges cost c^k such that $w^i(C^*) \leq b_i$ for $i = 1, \ldots, k-1$, and $s \in C^*$ ($t \notin C^*$).

1.1 Previous Work

Randomized contraction: Karger [7] gave an elegant randomized contraction algorithm that finds a global minimum cut with high probability. A consequence of its probabilistic analysis is a strongly polynomial bound on the number of (near-) optimal global minimum cuts. Karger and Stein [8] improve its running time using a recursive construction that carefully traded off the probability of success with the size of the recursive subproblems. Our work builds on these methods and extends them to budgeted versions of the global minimum cut problem.

Edge-budget constraints: While most budgeted versions of standard combinatorial optimization problems are NP-hard [4], Armon and Zwick [1] give an efficient strongly polynomial time algorithm for solving the minimum cut problem with a constant number k of edge-budget constraints. Their algorithm guesses the optimal value by performing a binary search using $O(\log n)$ calls to a subproblem called the *min-max cut problem*. Here, the goal is to find a cut $\bar{C}$ for which $\max_{i=1,\ldots,k} c^i(\bar{C})$ is minimized, *i.e.*, a cut $\bar{C}$ whose largest cost is the smallest possible. This problem is in turn reduced to enumerating all cuts that are at most at factor of k larger than the global minimum cut for a single cost function. Karger and Stein [8] show that every graph contains at most $O(n^{2k})$ such cuts. In order to enumerate them, Armon and Zwick use either the $O(mn^{2k})$ deterministic algorithm of Nagamochi et al. [11] or the $O(n^{2k} \log^2 n)$ randomized algorithm of Karger and Stein [8]. Thus, their approach leads to an $O(mn^{2k} \log n)$ time deterministic algorithm and an $O(n^{2k} \log^3 n)$ time randomized one. The minimum cut problem with edge-budget constraints may be of interest in itself but also arises as a subproblem in other fields, e.g., interdiction problems. Zenklusen [16] shows the link between the problem of maximally decreasing the optimal value of the global minimum cut by removing a limited set of edges and the minimum cut problem with a single edge-budget constraint.

Node-Budget Constraints: Armon and Zwick [1] consider the problem of finding a cut of minimum cost with at most b vertices on its smaller side. This problem corresponds to a special case of the single node-budget constraint ($k = 2$) with $w(v) = 1$ for all node $v \in V$. The authors reduce this problem to the problem of minimum cut with a single edge-budget constraint and give deterministic and randomized algorithms running in $O(mn^4 \log n)$ and $O(n^4 \log^3 n)$ times respectively. Goemans and Soto [5] consider the more general problem of minimizing a symmetric submodular functions (SSF) f over a family of sets $\mathcal{I}$ that are closed under inclusion. Note that the cut function over the node set of a graph $G = (V, E)$ is a SSF. Moreover, the family of all subset of nodes $X \subseteq V$ satisfying the node-budget constraints is a typical example of sets closed under inclusion. Goemans and Soto [5] extended Queyranne's algorithm [13] (which in turn is based on the work of Nagamochi and Ibaraki [10]) in order to enumerate all the $O(n)$ minimal minimizers using $O(n^3)$ oracle calls to function f and $\mathcal{I}$. In the particular case of graphs, their result implies that the minimum cut problem with node-budget constraints can be solved in $O(n^3)$ running time. Interestingly, Goemans and Soto's algorithm does not introduce any slowdown with respect to the running time of solving the global minimum cut problem.

Cardinality Constraints: Bruglieri et al. [2] study the version of minimum cut problem where the cardinality of the edge cut must be exactly the given bound k, or at least the given bound k, and show NP-hardness via reduction from MAX-CUT. The node-cardinality constrained version on the side containing a given source has been studied by Hayrapetyan et al. [6] under the name MINSBCC (Minimum-size bounded-capacity cut): their version bounds the cost of the cut and minimizes the node cardinality of the cut. They show NP-hardness of the problem on general graphs with uniform node weights and on trees (with non-uniform node weights), and provide bicriteria approximation algorithms with ratio $(\frac{1}{\lambda}, \frac{1}{1-\lambda})$ for any $0 < \lambda < 1$. The $s - t$ separating version of this unbalanced cut problem was studied by Li and Zhang [9], and by Zhang [17]. The node-cardinality constrained version of this problem generalizes the famous graph bisection problem. For the exact version of the problem where the side containing s must have exactly k nodes, an $O(\log n)$-approximation was given by Räcke [14].

1.2 Our contributions

The main contribution of our paper is to extend the Karger's randomized contraction algorithm [7] to handle node or edge budget constraints.

Edge-budget Constraints. The original randomized contraction algorithm for a single edge cost solves the global minimum cut problem by repeatedly picking a random edge with probability proportional to its cost and contracting it, until only two vertices remain. Karger shows that with high probability the cut formed by the edges joining these (super-)nodes is a minimum cut. The key ingredient in the proof of the success probability of Karger's algorithm is that the optimal value of the minimum cut problem is at most the cost of any cut formed by a singleton node. However, if budget constraints are added to the original problem, some of these singleton cuts may be infeasible (for the budget constraint) and hence may have a cost smaller than the optimal value of the budgeted problem. On the other hand, the cost of a feasible cut formed by a singleton node is larger than the optimal value but the current graph may contain few such nodes. Our main result uses new ideas to overcome this difficulty.

▶ **Theorem 1.** *For the global minimum cut problem with a single edge-budget constraint in a graph on n nodes, a randomized contraction algorithm returns any particular optimal solution in $O(n^3 \log^4 n \log\log n)$ time with probability $1 - \frac{1}{\Omega(n)}$.*

For the case of a single edge-budget constraint, our randomized contraction algorithm decides to contract, at each step, edges based on either the budget-cost function c^1 or the objective-cost function c^2, depending on whether the number of feasible cuts (obeying the budget) formed by the current singletons is sufficiently "large" or not. This modification is crucial to ensure the high success probability of returning at the end an optimal cut, and represents our main technical contribution.

Our final algorithm for this problem is presented in Section 2 and runs with high probability in $\tilde{O}(n^3)$ time. This result improves upon the current best running time of $\tilde{O}(n^4)$ given in [1]. As a byproduct of our analysis, we save a factor of $O(n)$ from the best-known upper bound on the number of optimal solutions of this problem given in [1].

In the general case, multiple edge-budget constraints make the problem harder because the number of infeasible cuts formed by a singleton may increase. With more than two budget constraints, a cut satisfying the i^{th} budget constraint may violate the j^{th} one. Therefore, even though the number of cuts formed by a singleton node satisfying the i^{th} budget constraint may be large (the property we used with a single budget constraint), few of them may satisfy all the budget constraints. Therefore, we need a different idea to tackle multiple budget constraints. For this case, we extend Karger's algorithm [7] differently by first sampling the cost function that is then used to randomly choose an edge to be contracted. Our final algorithm (Theorem 13) saves a logarithmic factor from the best-known running time of $O(n^{2k} \log^3 n)$ given in [1].

Node-budget Constraints. We derive in Section 3 faster randomized algorithms for finding global minimum cuts with a constant number of node budget constraints.

▶ **Theorem 2.** *For the global minimum cut problem with a constant number of node-budget constraint in a graph on n nodes, a randomized contraction algorithm returns any particular optimal solution in $O(n^2 \log n)$ time with probability $\Omega(1/\log n)$. Furthermore, all the optimal solutions can be computed with high probability in $O(n^2 \log^3 n)$ time.*

For this case, we use an observation similar to that of Goemans and Soto [5]: whenever the contraction produces two (node-budget) infeasible super-nodes, we merge them into one. Adding this idea to Karger's random contraction gives an algorithm with a randomized running time of $\tilde{O}(n^2)$ (Theorem 17). This considerably improves their current best running time of $O(n^3)$ [5] even though their algorithm is deterministic. As a byproduct, we show that the total number of distinct optimal global minimum cuts in the node-budget constrained case is also bounded by $\binom{n}{2}$ as in the non-budgeted case.

Our algorithm can be adapted to the node-budget constrained global min cut problem excluding a given sink $t \in V$ with the same running time and bound on number of optimal solutions (Theorem 18). In this case, the running time of our algorithm improves upon the previous deterministic running time of Goemans and Soto by a factor of $O(n)$.

Our results indicate that for the global minimum cut problem, the node-budget constraints are easier to handle than edge-budget ones, even though both are efficiently solvable. In contrast to the above results, we note that even the node-cardinality constrained global minimum cut problem containing a given source is strongly NP-hard using a reduction from graph bisection (Theorem 19).

Algorithm 1 Random edge contraction for a single edge-budget constraint.

Input: a simple graph $G = (V, E)$ with two nonnegative edge costs c^1, c^2, a bound b_1, and integer $q \geq 10$

Output: a cut $\emptyset \neq C^* \subset V$ minimizing cost c^2 subject to edge-budget constraint $c^1(\delta(C^*)) \leq b_1$

let $E_0 \leftarrow E$, $V_0 \leftarrow V$, $G_0 \leftarrow G$, $r \leftarrow 0$
while $|V_r| > 4$ **do**
  let $\hat{E}_r \leftarrow \emptyset$
  if $c^1(E_r) \leq \frac{b_1(|V_r|-1)}{6}$ **then**
    pick an edge $e \in E_r$ with probability $p(e) = \frac{c^2(e)}{c^2(E_r)}$ and add it to $\hat{E}_r$
  else
    for $i = 1$ to q **do**
      for each edge $e \in E_r \setminus \hat{E}_r$ **do**
        add e to $\hat{E}_r$ with probability $p'(e) = \frac{3c^1(e)}{b_1(|V_r|-1)}$
      end for
    end for
  end if
  if $\hat{E}_r \neq \emptyset$ **then**
    contract all the edges in $\hat{E}_r$ by merging their endpoints
    replace all resulting parallel edges $e_1, \ldots, e_p$ joining any pair of nodes $u, v \in V_r$ by a single edge e such that $c^h(e) = \sum_{i=1}^{p} c^h(e_i)$, $h = 1, 2$, and remove self-loops
  end if
  $r \leftarrow r + 1$
  let $G_r = (V_r, E_r)$ denote the resulting graph
end while
randomly partition the nodes in the final graph G' and return the cut C^* in G associated with this partition

2 Edge-budget constrained Global Minimum cuts

We discuss in Sections 2.1 and 2.2 our randomized algorithms for the single budget constraint and for multiple ones, respectively.

2.1 Single edge-budget constraint

The algorithm consists of two steps. The first reduces the graph by doing edge contractions until a minor graph G' with at most four nodes is obtained. In the second step, we randomly pick a cut in the resulting four-node graph.

Starting from $G_0 = (V_0, E_0) = G = (V, E)$, the first step of each iteration $r \geq 1$ consists of a possible reduction of graph $G_r = (V_r, E_r)$ to a graph $G_{r+1} = (V_{r+1}, E_{r+1})$ by contracting a sample edge set $\hat{E}_r \subseteq E_r$. The construction of $\hat{E}_r$ is performed as follows. First we set $\hat{E}_r = \emptyset$. Then two cases are considered: (i) If $c^1(E_r) \leq \frac{b_1(|V_r|-1)}{6}$, then we randomly pick an edge $e \in E_r$ with probability $p(e) = \frac{c^2(e)}{c^2(E_r)}$, and add it to $\hat{E}_r$. (ii) If this is not the case, then we add each edge $e \in E_r$ to $\hat{E}_r$ with probability $p'(e) = \frac{3c^1(e)}{b_1(|V_r|-1)}$. Note that the resulting sample edge set $\hat{E}_r$ may be empty. In order to boost the probability that $\hat{E}_r$ is non-empty, the process of random sampling is repeated q times, where q is a constant that will be specified later. If $\hat{E}_r \neq \emptyset$ at the end of the q trials, then we contract $\hat{E}_r$ and obtain a smaller graph $G_{r+1} = (V_{r+1}, E_{r+1})$. Otherwise, we set $G_{r+1} = G_r$.

An iteration r of the algorithm where condition $c^1(E_r) > \frac{b_1(|V_r|-1)}{6}$ holds and $\hat{E}_r = \emptyset$ is called *void*. Note that at most $|V| - 4$ non void iterations are performed but the total number of iterations may be large.

As a result of the edge contractions, parallel edges may join some pairs of vertices. Note that parallel edges are in the same cuts. Therefore, they can be replaced by a single edge with a cost equal to the sum of their costs. In contrast to Karger's algorithm [7], we need to consider only simple graphs at each step of Algorithm 1 in order to get the claimed running time (Lemma 10). This step is not essential for the analysis of Algorithm 1 but since it will be implemented recursively (Algorithm 2), $|E_r|$ must be bounded by $O(|V_r|^2)$ at each step r. All these details are summarized in Algorithm 1.

The following result gives a lower bound on the success probability that a particular optimal cut is returned by Algorithm 1.

▶ **Proposition 3.** *Any fixed optimal cut C^* is returned by Algorithm 1 with probability $\Omega\big(n^{-\frac{3}{1-\exp(-\frac{q}{2})}}\big)$.*

Our strategy to prove Proposition 3 is to handle separately the two cases in each iteration of the algorithm depending on whether $c^1(E_r) \leq \frac{b_1(|V_r|-1)}{6}$ or not. In the following two lemmas, we prove that the success probability of not contracting an edge in the optimal cut is at least $1 - \frac{3}{(|V_r|-1)(1-\exp(-\frac{q}{2}))}$ in each of these cases respectively.

Any edge in the current graph $G_r = (V_r, E_r)$ represents one or more edges in the original graph G. On the contrary, any edge in E is associated to at most one edge in E_r. Let E_r^{-1} denote the set of all the edges in E that are associated to the edges in E_r. For any set S of edges in E, let $E_r(S)$ denote, if any, the set of edges in E_r associated to the edges in S. An edge $e \in E$ has *survived* in graph G_r if $e \in E_r^{-1}$.

▶ **Lemma 4.** *Fix a particular optimal solution C^* and suppose that all the edges in $\delta(C^*)$ have survived in graph $G_r(V_r, E_r)$. If $c^1(E_r) \leq \frac{b_1(|V_r|-1)}{6}$, then the success probability of contracting an edge not in $E_r(\delta(C^*))$ is at least $1 - \frac{3}{|V_r|-1}$.*

Proof. Let $V_r^{\leq} \subseteq V_r$ denote the set of feasible nodes $v \in V_r$, *i.e.*, $c^1(\delta(\{v\})) \leq b_1$ for all $v \in V_r^{\leq}$. Observe that after replacing any parallel edges by a single one, the cost of any cut in the current graph G_r is the same as in the original graph G. Therefore, $c^2(\delta(C^*)) \leq c^2(\delta(\{v\}))$ for all node $v \in V_r^{\leq}$. Moreover, we have $\sum_{v \in V_r} c^1(\delta(\{v\})) = 2c^1(E_r) \leq \frac{b_1(|V_r|-1)}{3}$, and $\sum_{v \in V_r} c^1(\delta(\{v\})) \geq \sum_{v \in V_r \setminus V_r^{\leq}} c^1(\delta(\{v\})) > b_1|V_r \setminus V_r^{\leq}|$. Thus, $|V_r \setminus V_r^{\leq}| < \frac{b_1(|V_r|-1)}{3b_1} = \frac{|V_r|-1}{3}$, and hence,

$$|V_r^{\leq}| \geq \frac{2}{3}(|V_r| - 1). \tag{1}$$

Since all the edges in $\delta(C^*)$ have survived in G_r, we have $c^2(E_r(\delta(C^*)) = c^2(\delta(C^*))$. Therefore, the error probability of randomly picking an edge $e \in E_r(\delta(C^*))$ is

$$\begin{aligned} Pr(e \in E_r(\delta(C^*))) &= \frac{c^2(E_r(\delta(C^*))}{c^2(E_r)} = \frac{c^2(\delta(C^*))}{c^2(E_r)} \\ &\leq \frac{\sum_{v \in V_r^{\leq}} c^2(\delta(\{v\}))}{|V_r^{\leq}|c^2(E_r)} \leq \frac{\sum_{v \in V_r} c^2(\delta(\{v\}))}{|V_r^{\leq}|c^2(E_r)} \\ &= \frac{2}{|V_r^{\leq}|} \leq \frac{3}{|V_r| - 1} \text{ (by (1)).} \end{aligned}$$

◀

▶ **Lemma 5.** *Fix a particular optimal solution C^* and suppose that all the edges in $\delta(C^*)$ have survived in graph $G_r(V_r, E_r)$. If $c^1(E_r) > \frac{b_1(|V_r|-1)}{6}$, then $Pr(E_r(\delta(C^*)) \cap \hat{E}_r \neq \emptyset) \leq \frac{3}{|V_r|-1}$.*

Proof. Let $\hat{E}_r^i$ denote the set of edges in E_r added to $\hat{E}_r$ in trial $i = 1, \ldots, q$. Let μ denote the expected cardinality of $E_r(\delta(C^*)) \cap \hat{E}_r$. We have

$$\begin{aligned}\mu &= \sum_{i=1}^{q} \sum_{e \in E_r(\delta(C^*)) \cap \hat{E}_r^i} (1 - p'(e))^{i-1} p'(e) \leq \sum_{e \in E_r(\delta(C^*))} p'(e) \\ &= \sum_{e \in \delta(C^*)} \frac{3c^1(e)}{b_1(|V_r| - 1)} \text{ (all the edges in } \delta(C^*) \text{ have survived)} \\ &= \frac{3c^1(\delta(C^*))}{b_1(|V_r| - 1)} \leq \frac{3}{|V_r| - 1}.\end{aligned}$$

The last inequality comes from that C^* is a feasible cut, and thus $c^1(\delta(C^*)) \leq b_1$. Consequently,

$$Pr\big(E_r(\delta(C^*)) \cap \hat{E}_r \neq \emptyset\big) = Pr\big(|E_r(\delta(C^*)) \cap \hat{E}_r| \geq 1\big) \leq Pr\big(|E_r(\delta(C^*)) \cap \hat{E}_r| \geq \frac{|V_r| - 1}{3}\mu\big).$$

By Markov's inequality, $Pr\big(|E_r(\delta(C^*)) \cap \hat{E}_r| \geq \frac{|V_r|-1}{3}\mu\big) \leq \frac{3}{|V_r|-1}$ and thus

$$Pr(E_r(\delta(C^*)) \cap \hat{E}_r \neq \emptyset) \leq \frac{3}{|V_r| - 1}. \tag{2}$$

◀

▶ **Lemma 6.** *In graph $G_r = (V_r, E_r)$, if $c^1(E_r) > \frac{b_1(|V_r|-1)}{6}$, then $Pr(\hat{E}_r \neq \emptyset) > 1 - \exp(-\frac{q}{2})$.*

Proof. If $c^1(E_r) > \frac{b_1(|V_r|-1)}{6}$, then Algorithm 1 constructs $\hat{E}_r$ by randomly sampling all edges. Let F_r^i denote the event that the sample set $\hat{E}_r^i$ obtained during trial i is non-empty and $\bar{F}_r^i$ be the complementary event, $i = 1, \ldots, q$. We have

$$\begin{aligned}Pr(\hat{E}_r \neq \emptyset) &= Pr\big(\cup_{i=1}^{q} F_r^i\big) = 1 - Pr\big(\cap_{i=1}^{q} \bar{F}_r^i\big) \\ &= 1 - Pr(\bar{F}_r^q | \cap_{i=1}^{q-1} \bar{F}_r^i) Pr(\bar{F}_r^{q-1} | \cap_{i=1}^{q-2} \bar{F}_r^i) \cdots Pr(\bar{F}_r^2 | \bar{F}_r^1) Pr(\bar{F}_r^1) \\ &= 1 - \big(\Pi_{e \in E_r}(1 - p'(e))\big)^q = 1 - \big(\Pi_{e \in E_r}(1 - \frac{3c^1(e)}{b_1(|V_r| - 1)})\big)^q \\ &> 1 - \big(\Pi_{e \in E_r} \exp\big(-\frac{3c^1(e)}{b_1(|V_r| - 1)}\big)\big)^q = 1 - \big(\exp\big(-\sum_{e \in E_r} \frac{3c^1(e)}{b_1(|V_r| - 1)}\big)\big)^q \\ &= 1 - \exp\big(-\frac{3qc^1(E_r)}{b_1(|V_r| - 1)}\big) > 1 - \exp\big(-\frac{q}{2}\big).\end{aligned}$$

The last inequality comes from the fact that $c^1(E_r) > \frac{b_1(|V_r|-1)}{6}$. ◀

Proof of Proposition 3: If the condition of Lemma 4 holds, then the success probability at iteration r is $Pr(E_r(\delta(C^*)) \cap \hat{E}_r = \emptyset) \geq 1 - \frac{3}{|V_r|-1}$. Otherwise, we need to consider two cases depending on whether iteration r is void or not. In the former case, the success probability is $Pr(E_r(\delta(C^*)) \cap \hat{E}_r = \emptyset | \hat{E}_r = \emptyset) = 1$. Now if iteration r is non void, then by Lemma 5 we have $Pr(E_r(\delta(C^*)) \cap \hat{E}_r \neq \emptyset) \leq \frac{3}{|V_r|-1}$. In this case, we have

$$Pr(E_r(\delta(C^*)) \cap \hat{E}_r \neq \emptyset | \hat{E}_r \neq \emptyset) = \frac{Pr(E_r(\delta(C^*)) \cap \hat{E}_r \neq \emptyset)}{Pr(\hat{E}_r \neq \emptyset)} < \frac{3}{(|V_r| - 1)(1 - \exp(-\frac{q}{2}))},$$

where the last equality follows from Lemma 6. Therefore, the success probability satisfies $Pr(E_r(\delta(C^*)) \cap \hat{E}_r = \emptyset | \hat{E}_r \neq \emptyset) > 1 - \frac{3}{(|V_r|-1)(1-\exp(-\frac{q}{2}))}$.

Algorithm 2 Recursive random edge contraction for a single edge-budget constraint.

Input: a graph $G = (V, E)$ with two nonnegative edge costs c^1, c^2, a bound b_1, and $\alpha = \frac{3}{1-\exp(-\frac{q}{2})}$ for $q = \Omega(\log(\log^2 n))$ (this implies $\alpha = O(1)$)

Output: a cut $\emptyset \neq C^* \subset V$ minimizing cost c^2 subject to edge-budget constraint $c^1(\delta(C^*)) \leq b_1$

1: **if** $|V| \leq 6$ **then**
2: randomly partition the nodes in G and return the cut C^* defined by this partition
3: **else**
4: $t \leftarrow \lceil \frac{|V|}{\sqrt[\alpha]{2}} + 1 \rceil$
5: repeat twice
6: apply the while loop in Line 2 of Algorithm 1 and contract at each iteration r all the edges in $\hat{E}_r$ until obtaining a graph $G' = (V', E')$ with at most t nodes
7: recursively solve the problem on graph G'
8: return the best of the two cuts (obtained from the two different runs)
9: **end if**

By taking the product of all the success probabilities over all the iterations, the probability that all the edges in $\delta(C^*)$ have survived in the final graph G' is at least

$$(1-\frac{3/(1-\exp(-\frac{q}{2}))}{|V|-1})(1-\frac{3/(1-\exp(-\frac{q}{2}))}{|V|-2})\cdots(1-\frac{3/(1-\exp(-\frac{q}{2}))}{4}) = \Omega(|V|^{-\frac{3}{1-\exp(-\frac{q}{2})}}).$$

The probability of picking uniformly a cut in the final graph, formed by at most four nodes, is 2^{-4}. Therefore, multiplying both probabilities gives the desired result.

Using the same probabilistic argument to bound the number of minimum cuts as in Karger [7] and setting $q = O(\log(\log^2 n))$ in Proposition 3, we get the following result.

▶ **Corollary 7.** *The number of optimal solutions of the single edge-budget constrained global minimum cut problem is bounded by $O(n^3)$.*

The number of iterations required to have a nonempty sample set is a geometric random variable, which by Lemma 6, has an expected value bounded by $\frac{1}{1-\exp(-\frac{q}{2})}$. Observe that the $O(m) = O(n^2)$ running time of the random sampling is bottleneck in Algorithm 1. Therefore, the expected running time of the algorithm is $O(q \cdot n^3)$.

In order to amplify the success probability given by Proposition 3, one needs to perform $O(n^{\frac{3}{1-\exp(-\frac{q}{2})}} \log n)$ runs of Algorithm 1, which is excessive. Hence, we embed it in the recursive framework of Karger and Stein's [8] algorithm.

Our recursive algorithm can be represented using a binary tree where the root corresponds to graph G. And for every node of the tree, associated with some graph $H = (W, F)$, the algorithm constructs two graphs $H_1 = (W_1, F_1)$ and $H_2 = (W_2, F_2)$ obtained by performing two sequences of contractions as in Algorithm 1. However, in contrast to Algorithm 1, these contractions are stopped when the number of nodes in W is reduced by a factor $\sqrt[\alpha]{2}$, where $\alpha = \frac{3}{1-\exp(-\frac{q}{2})}$ and $q = \Omega(\log(\log^2 n))$. It is known that the depth of such tree is bounded by $\lfloor \log_{\sqrt[\alpha]{2}} n \rfloor$ and the number of leaves is at most

$$O(2^{\lfloor \log_{\sqrt[\alpha]{2}} n \rfloor}) \leq O(2^{\log_{\sqrt[\alpha]{2}} n}) = O(n^{\log_{\sqrt[\alpha]{2}} 2}) = O(n^{\alpha}) = O(n^{\frac{3}{1-\exp(-\frac{q}{2})}}) = O(n^3).$$

See Cormen et al. [3] for more details. This procedure is summarized in Algorithm 2.

The following result (restatement of Theorem 1) gives bounds on the probability of success and the running time of Algorithm 2.

▶ **Theorem 8.** *Algorithm 2 returns any particular optimal solution in* $O(n^3 \log^4 n \log\log n)$ *time with probability* $1 - \frac{1}{\Omega(n)}$.

The proof of Theorem 8 will be a consequence of the following lemmas (the proofs are omitted due to space limitations). The first one shows that Algorithm 2 has the same success probability as the recursive algorithm of Karger and Stein [8].

▶ **Lemma 9.** *A fixed optimal solution* C^* *is returned by Algorithm 2 with probability* $\Omega\big(\frac{1}{\log n}\big)$.

▶ **Lemma 10.** *For* $q = O(\log(\log^2 n))$, *the expected running time is* $O(n^3 \log n \log\log n)$.

Using the observation that the running time of Algorithm 2 can be analyzed as a sum of several sums of geometric random variables, we provide an upper bound that holds with high probability.

▶ **Lemma 11.** *The probability that the running time of Algorithm 2 exceeds* $O(n^3 \log^2 n \log\log n)$ *is bounded by* $O(1/n)$.

By Lemmas 9 and 10, a particular optimal solution C^* is returned with high probability by performing $O(\log^2 n)$ calls to Algorithm 2, with each call to this algorithm taking expected $O(n^3 \log n \log\log n)$ time. By using the same argument as in Lemma 11, the running times of all these calls is $O(n^3 \log^4 n \log\log n)$ with high probability. This shows Theorem 8.

2.2 Multiple edge-budget constraints

We consider in this section the more general case where we have a constant number k of edge-budget constraints. Note that if k is variable, the problem is strongly NP-hard [1]. In the case of a single edge-budget constraint, Lemmas 4 and 5 show that the edges of an optimal cut form a small fraction of all the edges. Algorithm 1 exploits this crucial property in order to return an optimal cut with high probability. If the condition of Lemma 4 holds, then the number of feasible cuts formed by a singleton node is large. With more than two budget constraints, a cut satisfying the i^{th} budget constraint may violate the j^{th} one. Therefore, even though the number of cuts formed by a singleton node satisfying the i^{th} budget constraint may be large, few of them may satisfy all the budget constraints. Therefore, we need a different idea to tackle the difficulties raised by multiple constraints.

The basic idea of the following algorithm is to repeat contracting randomly chosen edges until obtaining a graph formed by $2k$ nodes. At this point, the algorithm returns a cut uniformly chosen at random in this graph. The main difference with Algorithm 1 lies in the way how the random selection is done.

In graph $G_r = (V_r, E_r)$ obtained at iteration r of the algorithm, a node $v \in V_r$ is called *feasible* if the cut $\delta(\{v\})$ satisfies all the edge-budget constraints. Otherwise, it is called *infeasible*. Let V_r^i for $i = 1, \ldots, k-1$ denote a subset of infeasible nodes in V_r violating the edge-budget constraint associated to cost c^i and V_r^k denote the subset of feasible nodes in V_r. We partition the nodes in V_r by assigning all the feasible nodes to V_r^k and assigning arbitrary any infeasible node v to one of the subsets V_r^i such that $c^i(\delta(\{v\})) > b_i$. Let E_r^i denote the subset of edges in E_r incident to at least a node in V_r^i for $i = 1 \ldots, k$. We choose randomly a set V_r^i with probability $p_i = \frac{|V_r^i|}{|V_r|}$ and then pick an edge $e \in E_r^i$ with probability $\frac{c^i(e)}{c^i(E_r^i)}$ and contract it. This procedure is summarized in Algorithm 3.

The following result gives a lower bound on the success probability of Algorithm 3 following arguments similar to Proposition 3 (the proof is omitted due to space limitation).

▶ **Lemma 12.** *Algorithm 3 outputs any fixed optimal cut* C^* *with probability* $\Omega(n^{-2k})$.

Algorithm 3 Random edge contraction for the edge-budget constrained minimum cut problem.

Input: a graph $G = (V, E)$ with k nonnegative edges cost $c^1, \ldots, c^k$ and $k-1$ nonnegative bounds $b_1, \ldots, b_{k-1}$

Output: a cut $\emptyset \neq C^* \subset V$ minimizing edges cost c^k subject to the constraints $c^i(C^*) \leq b_i$, for $i = 1, \ldots, k-1$

let $E_1 \leftarrow E$, $V_1 \leftarrow V$, $G_1 \leftarrow G$, $r \leftarrow 1$
while $|V_r| > 2k$ **do**
  flip a biased coin and choose set E_r^i with probability $p_i = \frac{|V_r^i|}{|V_r|}$
  pick randomly an edge $e \in E_r^i$ with probability $p(e) = \frac{c^i(e)}{c^i(E_r^i)}$
  contract e by merging its vertices and removing self-loops
  $r \leftarrow r + 1$
  let $G_r = (V_r, E_r)$ denote the resulting graph
end while
randomly partition the nodes in the final graph and return the cut C^* in G associated to this partition

Note that the lower bound given in Lemma 12 is the same as the one given in [8, Theorem 8.5] for the success probability of computing a specific k-approximate cut, *i.e.* a cut within a multiplicative factor k of the minimum. Therefore, by embedding Algorithm 3 in the recursive algorithm of Karger and Stein, one can show the following result.

▶ **Theorem 13.** *Algorithm 3 returns all optimal solutions for the edge-budget constrained min cut problem with $k-1$ budgets in $O(n^{2k} \log^2 n)$ with high probability.*

3 Node-Constrained Cut Problems

3.1 Node Budget-constrained Global Minimum Cut Problem

We discuss in this section a randomized algorithm for the minimum cut problem with node-budget constraints based on an extension of Karger's randomized contraction algorithm [7]. The algorithm exploits an observation given by Goemans and Soto [5] for solving the problem of minimizing a SSF f over a family of sets $\mathcal{I}$ that are closed under inclusion over a ground set V. A typical example of such a family is the knapsack family: Given a weight function $w : V \to \mathbb{R}^+$, consider the family $\mathcal{I} = \{A \subseteq V : \sum_{v \in A} w(v) \leq 1\}$. Let us first briefly review Goemans and Soto's algorithm which is based on an extension of Queyranne's algorithm [13].

Queyranne gave a combinatorial algorithm for minimizing a SSF f by extending the deterministic minimum cut algorithm of Nagamochi and Ibaraki [10]. The basic idea of Queyranne's algorithm is to construct an ordering $(v_1, \ldots, v_n)$ of the elements of the ground set V such that $f(v_n) \leq f(X)$ for all $X \subset V$ that separates v_n and v_{n-1}. Note that the element v_1 may be chosen arbitrary in this algorithm. The ordered pair (v_{n-1}, v_n) is called a *pendant pair*. The algorithm stores $\{v_n\}$ as a candidate solution and merges v_n and v_{n-1}. The process continues until only two elements are left. The best among all the stored candidates is an optimal solution.

In order to handle the knapsack constraint, Goemans and Soto [5] construct first a new element v_1 obtained by merging all the infeasible elements of V (not in $\mathcal{I}$) and compute an ordering $(v_1, \ldots, v_r)$. The authors observed that as in Queyranne's algorithm [13], (v_{r-1}, v_r)

Algorithm 4 Random edge contraction for the node-budget constrained min cut problem.

Input: a graph $G = (V, E)$ with nonnegative edges cost c, nonnegative node weights w^i for $i = 1, \ldots, k-1$, and node budgets b^i for $i = 1, \ldots, k-1$

Output: a feasible cut $\emptyset \neq C^* \subset V$ with minimum cost

1: let $E_1 \leftarrow E$, $V_1 \leftarrow V$, $r \leftarrow 1$, $V^> \leftarrow \{v \in V | w^i(v) > b^i$ for some $i \in \{1, \ldots, k-1\}\}$, $G_1 \leftarrow G \odot V^>$, i.e. G with all nodes of $V^>$ merged into a single infeasible supernode.
2: **while** $|V_r| > 3$ **do**
3: choose an arbitrary edge $e \in E_r$ with probability $\frac{c(e)}{c(E_r)}$
4: contract e by merging its endpoints and removing self-loops
5: **if** there exists two supernodes v and v' in V_r that are infeasible **then**
6: merge v and v'
7: **end if**
8: $r \leftarrow r + 1$
9: let $G_r = (V_r, E_r)$ denote the resulting graph
10: **end while**
11: return a feasible cut C^* in the final graph G' with minimum cost

is still a *pendant pair*. Our approach uses the same idea but our starting point is the random contraction algorithm of Karger.

Denote a cut X or a supernode representing a cut *infeasible* if its shore exceeds any of the node budget constraints, i.e. $w^i(X) > b^i$ for some $i \in \{1, \ldots, k-1\}$. Algorithm 4 maintains at most one infeasible supernode (denoting the contraction of many vertices) at any time and repeatedly tries to contract a randomly chosen edge. After a random contraction if a new infeasible supernode is formed, it is merged with the previously existing infeasible supernode deterministically. This process continues until the final graph G_r formed by only three supernodes. At this point, the algorithm selects a feasible cut C^* in G_r with minimum cost and outputs it as a candidate optimal solution. The full algorithm is described in Algorithm 4.

If V_r contains at least two infeasible nodes then any feasible cut does not separate them. In this case, these supernodes are merged safely in Step 6. Otherwise, V_r contains at most one infeasible supernode and in this case, Algorithm 4 randomly contracts, in Step 4, an edge in E_r. The following results show that the algorithm always find a feasible cut in the final graph G' and returns any fixed optimal cut with high probability (the proofs are omitted due to space limitation).

▶ **Lemma 14.** *The final graph G' always contain a feasible cut.*

▶ **Lemma 15.** *Algorithm 4 outputs any fixed optimal cut C^* with probability $\Omega(n^{-2})$.*

Using the same probabilistic argument to bound the number of minimum cuts as in Karger [7], Lemma 15 implies the following result.

▶ **Corollary 16.** *The number of optimal solutions of the node-budget constrained global minimum cut problem is bounded by $\binom{n}{2}$.*

Note that Algorithm 4 has the same error probability and running time as the original contraction algorithm [8, Theorem 2.2]. Therefore, we can embed it in the sophisticated recursive algorithm [8, Section 4] in order to produce an optimal cut with the same success probability and the same running time as for the global minimum cut problem (without the budget constraints). Furthermore, similarly to [8, Theorem 4.4], by executing the recursive

algorithm $O(\log^2 n)$ times, all the optimal solutions can be computed with high probability. The following result (restatement of Theorem 2) summarizes the resulting running times.

▶ **Theorem 17.** *An optimal cut of the node-budget constrained global minimum cut problem on an n-node graph can be computed in $O(n^2 \log n)$ time with probability $\Omega(1/\log n)$. Furthermore, all the optimal solutions can be computed with high probability in $O(n^2 \log^3 n)$ time.*

3.2 Node Budget-constrained sink-excluding Global Minimum Cut Problem

It is not hard to adapt Algorithm 4 for the node-budget constrained global minimum cut problem excluding a given sink $t \in V$, where we have a set of $k-1$ node-weight budget constraints on the shore of the cut excluding t. We obtain the following result (the full algorithm description is given in the full paper).

▶ **Theorem 18.** *An optimal cut of the node-budget constrained global minimum cut problem excluding a given sink in an n-node graph can be computed in $O(n^2 \log n)$ time with probability $\Omega(1/\log n)$. Furthermore, all the optimal solutions can be computed with high probability in $O(n^2 \log^3 n)$ time.*

3.3 Node-cardinality constrained Source-including Min-cuts

In contrast to the sink-excluding case, we show that even the node-cardinality constrained minimum cut problem containing a given source is strongly NP-hard using a reduction from graph bisection. Note that Hayrapetyan et al. [6] study the version that bounds the edge costs of the cut and minimizes the node-cardinality of the cut, and show NP-hardness of that version via a reduction from max-clique. We provide a direct hardness proof for our version (omitted due to space limitation) by reducing from graph bisection.

▶ **Theorem 19.** *The node-cardinality constrained minimum cut containing a given source is strongly NP-hard.*

On the other hand, for the exact version of the problem where the side containing s must have exactly k nodes, an $O(\log n)$-approximation was given by Räcke [14] using his approach for the graph bisection problem.

4 Conclusion

Our results show that beyond the running time improvement, Karger's randomized contraction algorithm is sufficiently flexible to tackle efficiently budget constraints. An important open question is whether the exact algorithms of Nagamochi and Ibaraki [10] and Stoer and Wagner [15] can be extended in order to handle these budget constraints, since they are based on similar observations but have the potential to lead to better deterministic algorithms for the problems we study.

References

1 Amitai Armon and Uri Zwick. Multicriteria global minimum cuts. *Algorithmica*, 46(1):15–26, 2006.

2 Maurizio Bruglieri, Francesco Maffioli, and Matthias Ehrgott. Cardinality constrained minimum cut problems: complexity and algorithms. *Discrete Applied Mathematics*, 137(3):311–341, 2004.

3 Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition, 2001.

4 Matthias Ehrgott. *Multicriteria optimization*. Springer Science & Business Media, 2006.

5 Michel X. Goemans and José A. Soto. Algorithms for symmetric submodular function minimization under hereditary constraints and generalizations. *SIAM Journal on Discrete Mathematics*, 27(2):1123–1145, 2013.

6 Ara Hayrapetyan, David Kempe, Martin Pál, and Zoya Svitkina. Unbalanced graph cuts. In *Algorithms – ESA 2005: 13th Annual European Symposium, Proceedings*, pages 191–202, 2005.

7 D. R. Karger. Global min-cuts in RNC, and other ramifications of a simple min-cut algorithm. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA'93, pages 21–30, 1993.

8 D. R. Karger and C. Stein. A new approach to the minimum cut problem. *Journal of the ACM*, 43(4):601–640, 1996.

9 Angsheng Li and Peng Zhang. Unbalanced graph partitioning. In *Algorithms and Computation: 21st International Symposium, ISAAC 2010, Proceedings, Part I*, pages 218–229, 2010.

10 Hiroshi Nagamochi and Toshihide Ibaraki. Computing edge-connectivity in multigraphs and capacitated graphs. *SIAM Journal on Discrete Mathematics*, 5(1):54–66, 1992. doi:10.1137/0405004.

11 Hiroshi Nagamochi, Kazuhiro Nishimura, and Toshihide Ibaraki. Computing all small cuts in an undirected network. *SIAM Journal on Discrete Mathematics*, 10(3):469–481.

12 M. Padberg and G. Rinaldi. An efficient algorithm for the minimum capacity cut problem. *Mathematical Programming*, 47(1):19–36, 1990.

13 Maurice Queyranne. Minimizing symmetric submodular functions. *Mathematical Programming*, 82(1):3–12, 1998.

14 Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 255–264. ACM, 2008.

15 Mechthild Stoer and Frank Wagner. A simple min-cut algorithm. *Journal of the ACM*, 44(4):585–591, 1997.

16 Rico Zenklusen. Connectivity interdiction. *Operations Research Letters*, 42(6):450–454, 2014.

17 Peng Zhang. A new approximation algorithm for the unbalanced min s–t cut problem. *Theoretical Computer Science*, 609:658–665, 2016.