THE GOVERNMENT OF THE RUSSIAN FEDERATION
FEDERAL STATE EDUCATIONAL INSTITUTION OF HIGHER EDUCATION NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS

Faculty of Computer Science
Educational program "Mathematics and Computer Science"

Research project on the topic:

Speeding up Sherman's 2017 Algorithm for Multicommodity Flow

Completed by a student:

group AMI 214, 3 course Kuznetsov German Mikhalovich

Accepted by the project manager:

Bruno Frederik Bauwens
Associate professor
Faculty of Computer Science

Moscow 2024

Содержание

1 Abstract

This course work examines a 2017 article by Jonah Sherman [16], which describes a method for regularization of non-convex functions that allows us to find asymptotically fast solutions to some NP-hard problems with good accuracy. Applying area-convex regularization, article present a nearly-linear time approximation algorithm for solving matrix inequality systems $AX \leq B$ over right-stochastic matrices X . By combining that algorithm with existing work on preconditioning maximum-flow, article obtain a nearly-linear time approximation algorithm for maximum concurrent flow in undirected graphs: given an undirected, capacitated graph with m edges and k demand vectors, the algorithm takes $\widetilde{O}(\frac{mk}{\epsilon})$ time and outputs k flows routing the specified demands with total congestion at most $1 + \epsilon$ times optimal. The problem is that despite the small complexity, the algorithm works for a long time due to the large constant, in our course we will try to improve some reductions that will allow us to apply it in practical problems that require quick solutions

Keywords

area convexity, l-infinity regularization, multicommodity flow, saddle point problem

2 Report

2.1 Description of the subject area

Linear multicommodity flow problems (MCF) are linear programs (LPs) that can be characterized by a set of commodities and an underlying network. A commodity is a good that must be transported from one or more origin nodes to one or more destination nodes in the network. In practice these commodities might be telephone calls in a telecommunications network, packages in a distribution network, or airplanes in an airline flight network. Each commodity has a unique set of characteristics and the commodities are not interchangeable. That is, you cannot satisfy demand for one commodity with another commodity. The objective of the MCF problem is to flow a many commodities as possible through the network without exceeding arc capacities. In the context of the course work, we will consider using these algorithms to route signals between users, which gives a special case - a graph with edges of the same capacity and limitations on the running time of the algorithm.

2.2 Regularization

The first technical innovation is in the algorithm that finds saddle points when one of the variables ranges over stochastic matrices.Conjugate regularization is one of the most powerful existing tools for solving the general problem. The technique, implicitly used by the multiplicative weights method[3, 11] for the unit simplex case, is developed more generally by Nesterov[8]. In [16], Sherman uses a weaker condition to solve the saddle point problem, namely, takes a regularizer that helps to obtain area-convexity and not strictly convex as it was previously in Nesterov's works, which helps to go around l_∞ barier and obtain $\widetilde{O}(\frac{mk}{\epsilon})$ algoritm for multicommodity flow problem

2.3 Right-Stochastic Matrix Problems

After proving some area-convexity properties and investigating the convergence time AMO(approximate minimization oracle) Sherman provides a regularizer that satisfies the properties proven above and provides an algorithm(Approximate Minimizer for) that, applied to the stochastic matrix characterizing the graph, solves the MCF problem

2.4 Research results

This report contains proof of many facts that Sherman did not describe in his article, organizes and structures the transitions between tasks for a better understanding and improves the theoretical estimate of the algorithm's running time by about 3 times

3 Multicommodity Flow

Let $G = (V, E)$ be directed graph,and let $D \in \mathbb{R}^{V \times E}$ be the discrete divergence operator for G.

$$(D)_{vj} = \begin{cases} 1 & \text{if } v \text{ is start point for edge } j \\ -1 & \text{if } v \text{ is end point for edge } j \\ 0 & \text{else} \end{cases}$$

A single-commodity flow $f \in \mathbb{R}^E$ specifies a quantity $f_e \geq 0$ be transported along edge e, while $(Df)_v$ is the net quantity transported out of vertex v. A k-commodity flow $F \in \mathbb{R}^{E \times k}$ specifies quantities $F_{ej} \geq 0$ for each commodity j and edge e. Then, $(DF)_{vj}$ is the net quantity of commodity j transported out of v by F , if $DF = B$, we say F routes $B(B \in \mathbb{R}^{V \times k}$ - shows the difference between incoming and outgoing flows). A flow F is said to be feasible with respect to capacities $c \in \mathbb{R}^E$ if $c_e \geq \sum_j F_{ej}$ for each edge e. Letting $C \in \mathbb{R}^{E \times E}$ be diagonal with $C_{ee} = c_e$, a feasible flow F may be written as $F = CX$ for right-sub-stochastic X(matrix X is right-sub-stochastic if $\forall i, j \ X_{ij} \geq 0$ and $\forall e \sum_j X_{ej} \leq 1$), because $C^{-1}F$ is right-sub-stochastic.

3.1 Maximum-concurrent flow problem

In the maximum-concurrent flow problem, we are given $G = (V, E)$ with capacities $c \in \mathbb{R}^E$, demands $B \in \mathbb{R}^{V \times k}$, and we must feasibly route λB for $\lambda > 0$ as large as possible. If λ_{opt} is the true such maximum, then a flow F is a $(1 - \epsilon)$ -approximation to maximum-concurrent flow if it is feasible and $DF = \lambda B$ where $\lambda \geq (1 - \epsilon)\lambda_{opt}$ (That is, we are given information about the ratio of the quantity of each commodity goes out and enters the vertex from the outside, and we want find λ , such total transported commodity is max).

3.2 Duplex

In an undirected graph, there are two notions of feasibility, corresponding to whether edges are half-duplex(the sum of the absolute values of all flows passing through this edge e is bounded by c_e) or full-duplex (the sum of the values of all flows going in each direction passing through edge e is bounded by c_e).

Lemma: Suppose $[F^+|F^-]$ is $(1 - \epsilon)$ -optimal for the full-duplex problem with demands $[B| - B]$. Then, F^+ is $(1 - \epsilon)$ -optimal for the half-duplex problems with demands B.

3.3 Right-sub-stochastic to Right-stochastic

In this article, J.Sherman uses a solution for the Right-Stochastic matrix problem to solve the maximum-concurrent flow problem, but as we see now our problem is set by the right-sub-stochastic matrix, more precisely, for given D, C, B find max λ such $\exists X$ - right-sub-stochastic and

$$\begin{cases} DF = \lambda B \\ F = CX \end{cases}$$

so Sherman present how to go to right-stochastic matrix:

Let $b = Dc - B\mathbf{1}$, where $\mathbf{1}$ is the all-1 vector, c - capacity, and add an extra commodity with demands b, to obtain a new problem with demands $[B|b]$. We argue the original demands are feasible iff the new demands are routable by a flow where all edges use full capacity. Clearly, if $[F|f]$ is feasible and routes $[B|b]$, then F is feasible and routes B. On the other hand, if F feasibly routes B, then $f = c - F\mathbf{1}$ has $Df = D(c - F\mathbf{1}) = D - B\mathbf{1} = b$.That is, $[F|f]$ has $[F|f]\mathbf{1} = c$ and routes $[B|b] \implies$ task is reduces to find max λ, such $\exists$ right-stochastic $X : DCX = \lambda B$

3.4 From RSMP to MCP

In this section we show reduction from Right-stochastic matrix problem to max-concurrent problem.

$$(\Delta_k^m)^{\oplus} = \left\{ Y \in \mathbb{R}^{m \times k} : Y \geq 0, \sum_e \max_j Y_{ej} \leq 1 \right\}$$

$$\Delta_k^m = \left\{ X \in \mathbb{R}^{m \times k} : X \geq 0, \forall e \sum_j X_{ej} = 1 \right\}$$

$$\|X\|_{\infty \to \infty} = \max_{v : \|v\|_\infty = 1} \|Xv\|_\infty$$

Let we have algorithm which solves RSMP in fast time:

Thm. 1.7 Exist algorithm that for given $B \in \mathbb{R}^{n \times k}, C \in \mathbb{R}^{m \times k}, A \in \mathbb{R}^{n \times m}$ and $\|A\|_{\infty \to \infty} \leq 1$, takes $O(knnz(A)\epsilon^{-1})$ time and outputs $X \in \Delta_k^m, Y \in (\Delta_k^m)^{\oplus}$ such that,

$$\epsilon \geq \max_{X' \in \Delta_k^m, Y' \in \left(\Delta_k^m\right)^{\oplus}} \Phi(Y', X) - \Phi(Y, X')$$

where,

$$\Phi(Y, X) = tr[Y^T AX] - tr[C^T X] - tr[B^T Y]$$

Then we can apply thm 1.7 with C = 0, and got

$$\epsilon \geq \max_{X' \in \Delta_k^m, Y' \in \left(\Delta_k^m\right)^{\oplus}} tr[(Y')^T(AX - B)] - tr[Y^T(AX' - B)]$$

1) If $\exists Y \geq 0$ such $tr[Y^T(AX' - B)] > 0 \ \forall X' \in \Delta_k^m$ got (2) from corollary 1.8

2) Else algorithm output some $Y \geq 0$ for which $\exists X' : tr[Y^T(AX' - B)] \leq 0 \Rightarrow$

$$\max_{X' \in \Delta_k^m} -tr[Y^T(AX' - B)] \geq 0 \Rightarrow$$

$$\epsilon \geq \max_{Y' \in \left(\Delta_k^m\right)^{\oplus}} tr[(Y')^T(AX - B)]$$

Suppose that $\exists k$:

$$\sum_j max(0, (AX - B)_{kj}) > \epsilon$$

then if we take $(Y)_{ij} = \mathbf{1}[i = k]\mathbf{1}[(AX - B)_{ij} > 0]$

$$tr[(Y')^T(AX - B)] > \epsilon$$

$\Rightarrow$ contradiction $\Rightarrow$ sum of positive values in every row $(AX - B) \leq \epsilon \Rightarrow$
$\exists R \in \Delta_k^m$ such $(AX - B) \leq \epsilon R \Rightarrow$

Col. 1.8 Exist algorithm that for given $B \in \mathbb{R}^{n \times k}, A \in \mathbb{R}^{n \times m}$ and $\|A\|_{\infty \to \infty} \leq 1$, takes $O(knnz(A)\epsilon^{-1})$ time and outputs $X \in \Delta_k^m, Y \in (\Delta_k^m)^{\oplus}$ either
1)$Y \geq 0$ such $tr[Y^T(AX' - B)] > 0 \ \forall X' \in \Delta_k^m$
2)X such $\exists R \in \Delta_k^m$ such $(AX - B) \leq \epsilon R$

As follows from paragraph 3.4, our task is for given D, C, B find max λ such $\exists X$ - right-stochastic and

$$\begin{cases} DF = \lambda B \\ F = CX \end{cases}$$

We can reduce this problem using binary search for λ to the deciding problem : for given $D, C, \lambda B$

decide $\exists X$ - right-stochastic such $DCX = \lambda B$ which gives additional $O(\log(\lambda))$ complexity (further we will be interested in the deciding problem)

Thus, our task is reduced to deciding if there is right-stochastic X satisfying $DCX = B$. The latter implies $RDCX = RB$ for any R. We may use corollary 1.8 to approximate such equality problems by doubling the rows to have two $RDCX \leq RB$ and $-RDCX \leq -RB$.We can choose R such that knnz(RDC) is small with a slight loss of accuracy, which can significantly reduce the running time of the algorithm, but in the basic implementation R is unit matrix, so $RDCX \leq RB$ and $-RDCX \leq -RB \iff DCX = B$

4 Theorem 1.3 for MCF problem

Definition 1.1. A δ-approximate minimization oracle (δ-AMO) for $\phi : C \to \mathbb{R}$ takes input $a \in \mathcal{Z}^*$, and outputs $z \in C$ such that,

$$a \cdot z - \phi(z) + \delta \geq \sup_{z' \in C} a \cdot z' - \phi(z') =: \phi^*(a)$$

Definition 1.2. ϕ is area-convex with respect to J on a convex set C, iff, for all $x, y, z \in C$,

$$\phi\left(\frac{x+y+z}{3}\right) \leq \frac{1}{3}(\phi(x) + \phi(y) + \phi(z)) - \frac{1}{3\sqrt{3}}(z-y)\mathrm{J}(y-x)$$

Theorem 1.3. Let J be alternating, C compact-convex, and $\phi : C \to [-\rho, 0]$. Suppose ϕ is area-convex with respect to $2\sqrt{3}$ J on C, and Φ is a δ-AMO for ϕ. Define a sequence by $z(0) = 0$,

$$z(t+1) = z(t) + \tilde{\Phi}(\mathrm{J}z(t))$$

where

$$\tilde{\Phi}(a) = \Phi(a + 2\ \mathrm{J}\Phi(a))$$

Then, for all $t > 0$, we have $\frac{1}{t}z(t) \in C$ and,

$$\delta + \rho/t \geq \max_{z' \in C} z' \cdot \mathrm{J}(z(t)/t)$$

The proof of Theorem 1.3 is described quite well in Sherman's work[16], so its proof is omitted in this report.

Theorem 1.3 describes the algorithm which solving the RSMP and provides an estimate of the work time.

Let :

$$C = \{(w, Y, X) : w \in \Delta_n, Y \geq 0, Y_{vj} \leq w_v, X \in \Delta_n^m\}$$

$$\mathrm{J}(a, B, C) = (0, MC, -M^T B)$$

$$(w, Y, X) \cdot (a, B, C) = tr[w^T a] + tr[Y^T B] + tr[C^T X]$$

so if we take

$$M = \begin{array}{c} \\ n \\ k \end{array} \begin{array}{c} \begin{array}{cc} m & k \end{array} \\ \begin{pmatrix} A & B \\ C^T & 0 \end{pmatrix} \end{array}$$

$$Y' = \begin{array}{c} \\ n \\ k \end{array} \begin{array}{c} \begin{array}{c} k \end{array} \\ \begin{pmatrix} Y \\ E \end{pmatrix} \end{array}$$

$$X' = \begin{array}{c} \\ m \\ k \end{array} \begin{array}{c} \begin{array}{c} k \end{array} \\ \begin{pmatrix} X \\ E \end{pmatrix} \end{array}$$

C' defined like C but add E to every matrices X, Y as above

then by thm 1.3 if we find some ϕ in C' for which the conditions of the theorem are fulfilled and $\delta < \frac{\epsilon}{2}$ we got:

$$\delta + \rho/t \geq \max_{z' \in C} z' \cdot \mathrm{J}(z(t)/t)$$

choose $t > \frac{2\rho}{\epsilon}$

$$\epsilon \geq \max_{(w', Y', X') \in C'} (w', Y', X') \cdot \mathrm{J}(w, Y, X)$$

for $(z(t)/t) = (w, Y, X)$

$$\epsilon \geq \max_{(w', Y', X') \in C'} tr[Y'^T MX] - tr[X'^T M^T Y] = tr[Y'^T MX] - tr[Y^T MX']$$

using block matrix multiplication we get

$$\epsilon \geq \max_{(w', Y', X') \in C} tr[Y'^T AX] - tr[C^T X] - tr[B^T Y'] - tr[Y^T AX'] + tr[C^T X'] + tr[B^T Y]$$

which is exactly what was required in Theorem 1.7,therefore, by presenting the regularizer with

such properties, J.Sherman proves Theorem 1.7

5 δ-AMO for ϕ

In this section we present approximate minimizer for ϕ and bound time of its convergence.The main idea of which is alternating minimization (where variables split in two blocks and each step is block minimization)

$C_1 := \{(w, Y) : w \in \Delta_n, Y \geq 0, Y_{vj} \leq w_v\}$

$C := C_1 \oplus \Delta_k^m$

Algorithm 1: Approximate Minimizer for ϕ
Input: $a \in \mathbb{R}^n, B \in \mathbb{R}^{n\times k}, C \in \mathbb{R}^{m\times k}, \delta > 0$
Output: $(w, Y, X) \in C$ such that
$a \cdot w + \operatorname{tr}[B^*Y] + \operatorname{tr}[C^*X] - \phi(w, Y, X) \geq \phi^*(0, B, C) - \delta$
Initialize $X \in \Delta_k^m$ arbitrarily
repeat $O(\log \delta^{-1})$ times
$(w, Y) \leftarrow \arg\max_{(w,Y)\in C_1} w \cdot a + \operatorname{tr}[B^*Y] - \phi(w, Y, X)$
$X \leftarrow \arg\max_{X\in\Delta_k^m} \operatorname{tr}[C^*X] - \phi(w, Y, X)$
end

5.1 Maximizing X

Define functions as in article:

$$\operatorname{chop}(t) = \max(0, \min(1, t))$$

$$\operatorname{srect}(b, x) = \max_{t\in[0,1]} tb - \frac{x}{2}t^2 = \begin{cases} 0, & \text{if } b \leq 0 \\ \frac{b^2}{2x}, & \text{if } 0 < b < x \\ b - \frac{1}{2}x & \text{else} \end{cases}$$

$$\operatorname{expwts}(x)_i = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

Algorithm 2: Maximizing X
Input: $C \in \mathbb{R}^{m\times k}, (w, Y) \in C_1$
Output: $X \in \Delta_k^m$ maximizing $\operatorname{tr}[C^*X] - \phi(w, Y, X)$
$\bar{w} \leftarrow \overline{\mathrm{A}}^* w$
$Z_{vj} \leftarrow \alpha \frac{Y_{vj}^2}{2w_v}$
$\bar{Z} \leftarrow \overline{\mathrm{A}}^* Z$
$X_e \leftarrow \text{expwts}\left(\frac{C_e - \bar{Z}_e}{3\alpha \bar{w}_e}\right)$

Prove

Output: $X \in \Delta_k^m$ maximizing

$a \cdot w + \operatorname{tr}[B^*Y] + \operatorname{tr}[C^*X] - \phi(w, Y, X) \geq \phi^*(0, B, C) - \delta \iff$

Output: $X \in \Delta_k^m$ maximizing

$\operatorname{tr}[C^*X] - \phi(w, Y, X)$

differentiating by X:

$\sum_{e,j}(C_{ej} - \sum_v \alpha \overline{A}_{ve}(\frac{Y_{v,j}^2}{2w_v} + 3w_v \log(X_{ej}) + 3w_v))dX_{ej}$

Since $\forall e \sum_j X_{ej} = 1 \Rightarrow \forall i, j$

$C_{ej} - \sum_v \alpha \overline{A}_{ve}(\frac{Y_{v,j}^2}{2w_v} + 3w_v \log(X_{ej}) + 3w_v) = C_{ei} - \sum_v \alpha \overline{A}_{ve}(\frac{Y_{v,i}^2}{2w_v} + 3w_v \log(X_{ei}) + 3w_v)$

by $\sum_v \overline{A}_{ve} w_v = \overline{w_e}$

$C_{ej} - \sum_v \alpha \overline{A}_{ve}(\frac{Y_{v,j}^2}{2w_v}) + 3\alpha \overline{w_e} \log(X_{ej}) = C_{ei} - \sum_v \alpha \overline{A}_{ve}(\frac{Y_{v,i}^2}{2w_v}) + 3\alpha \overline{w_e} \log(X_{ei})$

by $Z_{vj} \leftarrow \alpha \frac{Y_{v,i}^2}{2w_v}$ and $\overline{Z} \leftarrow \overline{A}^* Z$

$C_{ej} - \overline{Z_{ej}} + 3\alpha \overline{w_e} \log(X_{ej}) = C_{ei} - \overline{Z_{ei}} + 3\alpha \overline{w_e} \log(X_{ei})$

$X_{ei} \exp(\left(\frac{C_{ei} - \bar{Z}_{ei}}{3\alpha \bar{w}_e}\right)) = X_{ej} \exp(\left(\frac{C_{ej} - \bar{Z}_{ej}}{3\alpha \bar{w}_e}\right)) \Rightarrow$

$X_e = \text{expwts}\left(\frac{C_e - \bar{Z}_e}{3\alpha \bar{w}_e}\right)$ -maximize function

5.2 Maximizing w, Y

Algorithm 3: Maximizing w, Y
Input: $a \in \mathbb{R}^n, B \in \mathbb{R}^{n\times k}, X \in \Delta_k^m$
Output: $(w, Y) \in C_1$ maximizing $a \cdot w + \operatorname{tr}[B^*Y] - \phi(w, Y, X)$
$\bar{X} \leftarrow \overline{\mathrm{A}}X$
$s_e \leftarrow \sum_j X_{ej} \log X_{ej}$
$\bar{s} \leftarrow \overline{\mathrm{A}}s$
$b_v \leftarrow \sum_j \operatorname{srect}\left(B_{vj}, \alpha\bar{X}_{vj}\right)$
$w \leftarrow \operatorname{expwts}\left(\frac{a+b-3\alpha\bar{s}}{\alpha\beta}\right)$
$Y_{vj} \leftarrow w_v \operatorname{chop}\left(B_{vj}/\alpha\bar{X}_{vj}\right)$

Prove

Output: $(w, Y) \in C_1$ maximizing $\quad f(w, Y) = a \cdot w + \operatorname{tr}[B^*Y] + \operatorname{tr}[C^*X] - \phi(w, Y, X) \geq \phi^*(0, B, C) - \delta \iff$
Output: $(w, Y) \in C_1$ maximizing $\quad a \cdot w + \operatorname{tr}[B^*Y] - \phi(w, Y, X)$

differentiating $f(w, Y)$ by w:

$\sum_v (a_v + \alpha \sum_{e,j} \overline{A}_{ve} X_{ej} \frac{Y_{v,j}^2}{2w_v^2} + 3\alpha \sum_{e,j} \overline{A}_{ve} X_{ej} \log X_{ej} + \alpha\beta \log w_v + \alpha\beta) dw_v$

applying $\bar{X} \leftarrow \overline{\mathrm{A}}X$, $s_e \leftarrow \sum_j X_{ej} \log X_{ej}$ and $\bar{s} \leftarrow \overline{\mathrm{A}}s$ get

1 $\sum_v (a_v + \alpha \sum_j \overline{X}_{vj} \frac{Y_{v,j}^2}{2w_v^2} + 3\alpha\bar{s} + \alpha\beta \log w_v + \alpha\beta) dw_v$

differentiating $f(w, Y)$ by Y:

$\sum_{v,j} (B_{vj} - \alpha\overline{X}_{vj} \frac{Y_{vj}}{w_v}) dY_{vj}$

since $\forall v, j \; 0 \leq Y_{vj} \leq w_v$ and $\alpha, \overline{X}_{vj}, w_v > 0$

$$Y_{vj} = \begin{cases} 0, & \text{if } B_{vj} \leq 0 \\ \frac{B_{vj} w_v}{\alpha \overline{X}_{vj}}, & \text{if } 0 < B_{vj} < \alpha\overline{X}_{vj} \\ w_v & \text{else} \end{cases}$$

replace Y_{vj} in 1:

$\sum_v (a_v + \alpha \sum_j \text{srect}\left(B_{vj}, \alpha \bar{X}_{vj}\right) + 3\alpha\overline{s_v} + \alpha\beta \log w_v + \alpha\beta) dw_v$

$\sum_j \text{srect}\left(B_{vj}, \alpha \bar{X}_{vj}\right) = b_v$ and $\sum_v w_v = 1 \Rightarrow \forall v, l$

$a_v + \alpha b_v + 3\alpha\overline{s_v} + \alpha\beta \log w_v + \alpha\beta = a_l + \alpha b_l + 3\alpha\overline{s_l} + \alpha\beta \log w_v + \alpha\beta \Rightarrow$

$w = \text{expwts}\left(\frac{a+b-3\alpha\bar{s}}{\alpha\beta}\right)$ - maximize function $\Rightarrow$

$Y_{ij} = max(0, min(1, \frac{B_{vj} w_v}{\alpha \bar{X}_{vj}}) = w_v \, \text{chop}\left(B_{vj}/\alpha \bar{X}_{vj}\right)$ - maximize function

5.3 Convergence of δ-AMO

In this paragraph, we will show a more accurate convergence estimate than in the original, which will reduce the running time of the algorithm

For the convergence proof,we use the fact that w does not change much.

$s_e = \sum_j X_{ej} \log X_{ej}$, since $\sum_j X_{ej} = 1$ get $-\log k \leq s_e \leq 0$

$\bar{s}_e = As_e$, $\|\bar{A}\|_{\infty\to\infty} \leq 1 \Rightarrow -\log k \leq \bar{s}_e \leq 0$

Noting:

$\operatorname{srect}(t,0) - \frac{x}{2} \leq \operatorname{srect}(x,t) \leq \operatorname{srect}(t,0)$, we can sum over j

$-\frac{\alpha}{2} \leq \sum_j \operatorname{srect}(B_{vj}, \alpha\bar{X}_{vj}) - \sum_j \operatorname{srect}(B_{vj}, 0) \leq 0 \Rightarrow$

$$\frac{\max_i w_i}{\min_i w_i} = \frac{\max_i \exp\left(\frac{a_i+b_i-3\alpha\bar{s}_i}{\alpha\beta}\right)}{\min_i \exp\left(\frac{a_i+b_i-3\alpha\bar{s}_i}{\alpha\beta}\right)} \leq \exp\left(\frac{0.5\alpha + 3\alpha\log k}{\alpha\beta}\right) =$$

$$=\exp\left(\frac{0.5+3\log k}{27(2+\log k)^2}\right) \leq \exp\left(\frac{1}{9(2+\log k)}\right) \leq 1.06 = c$$

We use that $a = 0$ in δ-AMO calls $\Rightarrow$

Lemma 3.6(more accurate) Let $X, X' \in \Delta_k^m$.Let (w, Y) minimize $\eta(w, Y, X)$ and (w', Y') minimize $\eta(w', Y', X')$.Then,

$$\frac{w}{c} \leq w' \leq cw$$

Let w, Y be given and fixed for the rest of the subsection, let X minimize $\eta(w, Y, X)$, and let w', Y' minimize $\eta(w', Y', X)$. Define,

$$\Pi(w'', Y'', X'') := (w'', Y'', 0)$$

$$\tilde{C}_1 := \left\{(w'', Y'') \in C_1 : \frac{1}{c}w \leq w'' \leq cw\right\}$$

$$\tilde{C}_2 := \left\{X'' \in \Delta_k^m : X'' \geq \frac{1}{2}X\right\}$$

$$M := \frac{1}{3c^2}\Pi Q(w, Y, X)\Pi$$

Note the definition of Q (article (14)), together with $\tilde{C}_i$ imply that for $(w'', Y'') \in \tilde{C}_1, X'' \in \tilde{C}_2$, we have,

$$\Pi Q(w'', Y'', X'')\,\Pi \geq \frac{1}{2c}\Pi Q(w, Y, X)\Pi \geq \frac{1}{2c^2}\Pi Q(w'', Y'', X)\,\Pi$$

Then, by (article lemma 3.4),

$$\begin{aligned}
\mathrm{d}^2\eta\left(w'', Y'', X''\right) &\geq \frac{1}{3}Q\left(w'', Y'', X''\right) \\
&\geq \frac{1}{3}\Pi Q\left(w'', Y'', X''\right)\Pi \\
&\geq M \\
&\geq \frac{1}{6c^2}\Pi Q\left(w'', Y'', X\right)\Pi \\
&\geq \frac{1}{12c^2}\Pi \mathrm{d}^2\eta\left(w'', Y'', X\right)\Pi
\end{aligned}$$

Lemma 3.6(more accurate) establishes that $(w_{\mathrm{opt}}, Y_{\mathrm{opt}}) \in \tilde{C}_1$, but have no guarantee about X_{opt}. However, we do have $\frac{1}{2}\left(X_{\mathrm{opt}} + X\right) \in \tilde{C}_2$. Thus, for the convex combination $(w'', Y'', X'') = \frac{1}{2}\left(w_{\mathrm{opt}} + w, Y_{\mathrm{opt}} + Y, X_{\mathrm{opt}} + X\right)$, we have $\eta\left(w'', Y'', X''\right) \leq \frac{1}{2}\eta(w, Y, X)$, and Beck's theorem yields,

$$\eta(w, Y, X) - \min_{(w', Y') \in C_1} \eta\left(w', Y', X\right) \geq \frac{1}{24c^2}\eta(w, Y, X)$$

That is, each (w, Y) block-minimization reduces η by a constant factor around $\frac{26}{27}$ instead of $\frac{95}{96}$.

6 Conclusion

This article describes the multicommodity flow algorithm which solves max-concurrent-flow problem and takes by theorem 1.3 $\rho\epsilon^{-1} \simeq \frac{27*18*\log(k)^2*\log(n)}{\epsilon}$ steps of stepping lemma[16], each of step require $\delta - AMO$ ($\delta - AMO$ contains cycle with block minimization, each of which can be calculated in $O(knnz(A))$, and cycle must have at least $\log_{\frac{27}{26}}(\frac{1}{\epsilon}) \simeq 27\log(\frac{1}{\epsilon}))$ call , so overall algorithm time is $O(knnz(A)\epsilon^{-1}\log(n)\log(k)^2\log(\frac{1}{\epsilon}))$.Because we made the theoretical bound more tight (5.3) it speeds up the algorithm exactly in 3 times.

REFERENCES

[1] Amir Beck. 2015. On the convergence of alternating minimization for convex programming with applications to iteratively reweighted least squares and decomposition schemes. SIAM Journal on Optimization 25, 1 (2015), 185–209.

[2] Hui Han Chin, Aleksander Madry, Gary L Miller, and Richard Peng. 2013. Runtime guarantees for regression problems. In Proceedings of the 4th conference on Innovations in Theoretical Computer Science. ACM, 269–282.

[3] Yoav Freund and Robert E Schapire. 1999. Adaptive game playing using multiplicative weights. Games and Economic Behavior 29, 1 (1999), 79–103.

[4] Jonathan A Kelner, Yin Tat Lee, Lorenzo Orecchia, and Aaron Sidford. 2014. An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, 217–226.

[5] Jonathan A Kelner, Gary L Miller, and Richard Peng. 2012. Faster approximate multicommodi flow using quadratically coupled flows. In Proceedings of the forty-fourth annual ACM symposium on Theory of computing. ACM, 1–18.

[6] GM Korpelevich. 1977. Extragradient Method for Finding Saddle Points and Other Problems. Matekon 13, 4 (1977), 35–49.

[7] Arkadi Nemirovski. 2004. Prox-method with rate of convergence O (1/t) for variational inequalities with Lipschitz continuous monotone operators and smooth convex-concave saddle point problems. SIAM Journal on Optimization 15, 1 (2004), 229–251.

[8] Yu Nesterov. 2005. Smooth minimization of non-smooth functions. Mathematical programming 103, 1 (2005), 127–152.

[9] Yurii Nesterov. 2007. Dual extrapolation and its applications to solving variational inequalities and related problems. Mathematical Programming 109, 2-3 (2007), 319–344.

[10] Richard Peng. 2016. Approximate undirected maximum flows in O (m polylog (n)) time. In Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms. SIAM, 1862–1867.

[11] Serge A. Plotkin, David B. Shmoys, and Eva Tardos. 1995. Fast approximation algorithms for fractional packing and covering problems. Mathematics of Operations Research 20 (1995), 257–301.

[12] Ralph Tyrell Rockafellar. 2015. Convex analysis. Princeton university press.

[13] Jonah Sherman. 2013. Nearly maximum flows in nearly linear time. In Foundations of

Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on. IEEE, 263–269.

[14] Jonah Sherman. 2017. Generalized Preconditioning and Network Flow. In Proceedings of the Twenty-Eigth Annual ACM-SIAM Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics, to appear.

[15] Daniel A. Spielman and Shang-Hua Teng. 2006. Nearly-Linear Time Algorithms for Preconditioning and Solving Symmetric, Diagonally Dominant Linear Systems. CoRR abs/cs/0607105 (2006).

[16] Jonah Sherman. 2017. Area-Convexity, l Regularization, and Undirected Multicommodity Flow. In Proceedings of 49th Annual ACM SIGACT Symposium on the Theory of Computing, Montreal, Canada, June 2017 (STOC'17)