Optimal Traffic Engineering Via Newton's Method

Dahai Xu
dahaixu@research.att.com
AT&T Labs - Research, New Jersey, USA

***Abstract*—Recently, it has been proven that the minimum-cost multicommodity flow can be realized by a link-state routing protocol, PEFT, using uneven traffic splitting [1]. The achievement is due to solving a new convex optimization problem, Network Entropy Maximization (NEM), by a gradient descent algorithm. However, gradient descent algorithms are notorious for slow convergence to optimality whereas Newton's method usually converges much faster. Solving NEM by Newton's method is neither trivial nor standard because of the infinite number of variables NEM has. In this paper, we develop efficient Newton methods for the NEM problem.**

Keywords: Interior gateway protocol, optimal traffic engineering, routing, network entropy maximization, Newton's method.

I. Introduction

The classical minimum-cost multicommodity flow problem is to transport multiple commodity flows while minimizing a convex objective function of the aggregate flow on every link. Its optimal solution can be found quickly through convex optimization. It has important applications in traffic engineering, or congestion control in IP networks, where the network operators try to balance the traffic within the network by assigning a high penalty cost on those congested links.

Traffic engineering for link-state routing protocols has attracted a lot of interest [2]–[6] since the pioneering work of Fortz and Thorup [7]. In a link-state routing protocol, the network-management system computes a set of link weights through a periodic and centralized optimization; then each router uses the link weights to decide traffic splitting fractions for every destination among its outgoing links [1]. It had confused the community for a while as to whether a link-state routing protocol can realize the optimal solution of the minimum-cost multicommodity flow problem.

For the first time, Xu et al. [1] show that a new link-state routing protocol "PEFT" can achieve optimality using uneven traffic splitting across outgoing links. The accomplishment is due to solving a new convex optimization problem, Network Entropy Maximization (NEM), by a gradient descent algorithm. In this paper, we propose solving NEM by Newton's method (scaled gradient descent) since Newton's method usually converges much faster and makes choosing appropriate step sizes easier. Note that the NEM problem has an infinite number of variables, which makes the extension even more challenging.

The rest of the paper is organized as follows. Background on the minimum-cost multicommodity flow problem and the theory of Network Entropy Maximization is introduced in Sec. II. Newton's method for NEM is discussed in Sec. III. We numerically compare the performance of solving NEM by gradient descent algorithms and Newton's methods in Sec. IV and conclude the paper in Sec. V.

The key notation used in this paper is shown in Table I.

II. Background on Optimal Traffic Engineering and NEM

In this section, we summarize the modeling and gradient descent method for the NEM problem proposed in [1] to realize optimal traffic engineering.

A. Optimal Traffic Engineering Via Multicommodity Flow

Consider a wireline network as a directed graph $G = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V}$ is the set of nodes (where $N = |\mathbb{V}|$), $\mathbb{E}$ is the set of links (where $E = |\mathbb{E}|$), and link (u, v) has capacity $c_{u,v}$. The offered traffic is represented by a traffic matrix $D(s, t)$ with source-destination pairs indexed by (s, t).

Define $f_{u,v}$ to be the load on link (u, v), which depends on routing protocols and traffic demands. $\Phi(\{f_{u,v}, c_{u,v}\})$ is an increasing and convex objective function of $\{f_{u,v}\}$, which could be the maximum link utilization, or a piecewise-linear approximation of the M/M/1 delay formula [7], etc.

Then we have the following classical convex optimization problem, the min-cost multicommodity problem (1), where the flow destined to a single destination is treated as a commodity, and $f^t_{u,v}$ is the amount of flow on link (u, v) destined to node t.

COMMODITY:

$$\begin{aligned}
\min \quad & \Phi(\{f_{u,v}, c_{u,v}\}) && \text{(1a)}\\
\text{s.t.} \quad & \sum_{v:(s,v)\in\mathbb{E}} f^t_{s,v} - \sum_{u:(u,s)\in\mathbb{E}} f^t_{u,s} = D(s,t), \forall s \neq t && \text{(1b)}\\
& f_{u,v} \triangleq \sum_{t\in\mathbb{V}} f^t_{u,v} \leq c_{u,v}, \forall (u,v) && \text{(1c)}\\
\text{vars.} \quad & f^t_{u,v}, f_{u,v} \geq 0. && \text{(1d)}
\end{aligned}$$

Necessary Capacity

Given the traffic matrix and a convex objective function, the solution to the above COMMODITY problem (1) represents the optimal distribution of traffic. The solution flow, $\widetilde{c}_{u,v} \triangleq f_{u,v}$ on each link (u, v) (or $\widetilde{\boldsymbol{c}}$ as a vector), is called the *necessary capacity* to achieve the optimal objective [1].

The PEFT protocol proposed in [1] is the first and the only link-state routing protocol known so far to realize the optimal distribution of traffic. It is derived from a new optimization problem, Network Entropy Maximization.

TABLE I
SUMMARY OF KEY NOTATION

Notation	Meaning
$D(s,t)$	Traffic demand from source s to destination t
$\widetilde{c}_{u,v}$	Necessary capacity of link (u,v)
$w_{u,v}$	Weight assigned to link (u,v)
$f_{u,v}$	Flow on link (u,v)
$f^t_{u,v}$	Flow on link (u,v) destined to node t
f^t_u	Total flow (destined to t) at u
$\eta^{s,t}_u$	Total flow at u for a unit traffic demand from s to t
$\psi^t_{u,v}$	Splitting fraction (destined to t) on link (u,v)

B. Network Entropy Maximization

Define $P_{s,t}$ to be the set of paths from s to t (repeated nodes are allowed), and $x^i_{s,t}$ to be the probability (fraction) of forwarding a packet of demand $D(s,t)$ along the i-th path ($P^i_{s,t}$). Define $K^{(u,v)}_{P^i_{s,t}}$ to be the number of times $P^i_{s,t}$ passes through link (u,v); $P^i_{s,t}$ can contain cycles.

The Network Entropy Maximization (**NEM**) problem under the necessary capacity constraints is defined in [1] as follows [1]:

NEM:

$$\max \quad \sum_{s,t}\left(D(s,t)\sum_{i\in P_{s,t}} -x^i_{s,t}\log x^i_{s,t}\right) \tag{2a}$$

$$\text{s.t.} \quad \sum_{s,t,i} D(s,t)K^{(u,v)}_{P^i_{s,t}} x^i_{s,t} \le \widetilde{c}_{u,v}, \forall (u,v) \tag{2b}$$

$$\sum_i x^i_{s,t} = 1, \forall s,t \tag{2c}$$

$$\text{vars.} \quad x^i_{s,t} \ge 0. \tag{2d}$$

There exist globally optimal solutions to NEM, which is also a convex optimization problem [1].

C. Solving the NEM Dual By Gradient Descent

Though the NEM problem has an infinite number of variables, the values of the dual variables for constraints (2b), $\lambda_{u,v}$ for link (u,v) (or $\boldsymbol{\lambda}$ as a vector), are sufficient for a link-state routing protocol. Q, the Lagrange dual problem of NEM, can be solved by a gradient descent algorithm [1], which is shown as follows for iterations indexed by q.

$$\boldsymbol{\lambda}(q+1) = \left[\boldsymbol{\lambda}(q) - \alpha(q)\nabla Q(\boldsymbol{\lambda}(q))\right]^+, \tag{3}$$

where $\alpha(q) > 0$ is the step size and $\nabla Q(\boldsymbol{\lambda}(q))$ is the gradient. At the optimal solution of NEM, we have

$$x^i_{s,t} = e^{-\left(\sum_{(u,v)} K^{(u,v)}_{P^i_{s,t}}\lambda_{u,v} + \frac{\mu_{s,t}}{D(s,t)} + 1\right)}, \tag{4}$$

[1] $\sum_{s,t,i:(u,v)\in P^i_{s,t}} D(s,t)x^i_{s,t}(q)$ is used in [1] instead of $\sum_{s,t,i} D(s,t)K^{(u,v)}_{P^i_{s,t}} x^i_{s,t}$.

where $\mu_{s,t}$ is the dual variable for constraint (2c). From [8], we have

$$\begin{aligned}&\tfrac{\partial Q}{\partial \lambda_{u,v}}(q)\\ &= \widetilde{c}_{u,v} - \textstyle\sum_{s,t,i} D(s,t)K^{(u,v)}_{P^i_{s,t}} x^i_{s,t}(q)\\ &= \widetilde{c}_{u,v} - f_{u,v}(q),\end{aligned} \tag{5}$$

and from (4), we have

$$\frac{x^i_{s,t}}{x^j_{s,t}} = \frac{e^{-\sum_{(u,v)} K^{(u,v)}_{P^i_{s,t}}\lambda_{u,v}}}{e^{-\sum_{(u,v)} K^{(u,v)}_{P^j_{s,t}}\lambda_{u,v}}}. \tag{6}$$

Then $\lambda_{u,v}$ can be used as the weight $w_{u,v}$ for link (u,v), and the probability of using path $P^i_{s,t}$ is inversely proportional to the exponential of its path length. Since (6) has no factor of $\mu^*_{s,t}$, an intermediate router can ignore the source of the packet in forwarding [1].

D. A New Link-State Routing Protocol: PEFT

Eq. (6) results in a new link-state routing protocol, Penalizing Exponential Flow-spliTting (**PEFT**), which satisfies (7), where $p^i_{u,t}$ is the path length of the ith path (i.e., the sum of $w_{u,v}$ of the links along the path).

$$\text{PEFT:} \qquad x^i_{u,t} = \frac{e^{-p^i_{u,t}}}{\sum_j e^{-p^j_{u,t}}}. \tag{7}$$

To forward a packet destined to t by a link-state routing protocol, each router u needs to know the traffic splitting fraction $\psi^t_{u,v}$ on outgoing link (u,v). Based on a complete view of the topology and link weights, a router can compute the distance d^t_u from any node u to node t, and the excess length $h^t_{u,v} \triangleq d^t_v + w_{u,v} - d^t_u$ over the distance.

From (7), more traffic should be sent along an outgoing link used by more paths and the paths should be treated differently based on their path lengths. Υ^t_u is defined as the "equivalent number" of shortest paths from node u to destination t as shown in equation (8a); let $\Upsilon^t_t \triangleq 1$. Υ^t_u also can be computed recursively by solving linear equations (8b) [1].

$$\Upsilon^t_u \triangleq \sum_{i\in P_{u,t}} e^{-\left(p^i_{u,t} - d^t_u\right)} \tag{8a}$$

$$\begin{aligned}&= \sum_{(u,v)\in\mathbb{E}}\left(\sum_{j\in P_{v,t}} e^{-\left(p^j_{v,t} + w_{u,v} - d^t_u - d^t_v + d^t_v\right)}\right)\\ &= \sum_{(u,v)\in\mathbb{E}}\left(e^{-\left(d^t_v + w_{u,v} - d^t_u\right)}\sum_{j\in P_{v,t}} e^{-\left(p^j_{v,t} - d^t_v\right)}\right)\\ &= \sum_{(u,v)\in\mathbb{E}}\left(e^{-h^t_{u,v}}\Upsilon^t_v\right)\end{aligned} \tag{8b}$$

Then

$$\psi^t_{u,v} = \frac{e^{-h^t_{u,v}}\Upsilon^t_v}{\Upsilon^t_u}. \tag{9}$$

Let f^t_u denote the total flow destined to t which passes through node u or originates at u. Then $f^t_{u,v}$, the total outgoing

flow of traffic (destined to t) traversing link (u,v), can be computed as follows:

$$f^t_{u,v} = \psi^t_{u,v} f^t_u. \tag{10}$$

For $\{D(s,t)\}$, the given input traffic demand, and $\{w_{u,v}\}$, a set of link weights, f^t_u can be computed by solving the following linear equations:

$$f^t_s - \sum_{y:(y,s)\in\mathbb{E}} \psi^t_{y,s} f^t_y = D(s,t). \tag{11}$$

III. Newton's Method to Solve NEM

In this section, we present Newton's method to solve the dual problem of NEM. The main challenge is computing the Hessian matrix given that NEM has an infinite number of variables.

A. Solving NEM Dual By Newton's Method

In Newton's method, the gradient is scaled by the inverse of $\nabla^2 Q(\boldsymbol{\lambda}(q))$, the Hessian matrix for the q-th iteration.

$$\boldsymbol{\lambda}(q+1) = \left[\boldsymbol{\lambda}(q) - \nabla^2 Q(\boldsymbol{\lambda}(q))^{-1} \nabla Q(\boldsymbol{\lambda}(q))\right]^+. \tag{12}$$

From (4), we have

$$\begin{aligned} \frac{\partial x^i_{s,t}}{\partial \lambda_{u,v}} &= -K^{(u,v)}_{P^i_{s,t}} e^{-\left(\sum_{(u,v)} K^{(u,v)}_{P^i_{s,t}} \lambda_{u,v} + \frac{\mu_{s,t}}{D(s,t)} + 1\right)} \\ &= -K^{(u,v)}_{P^i_{s,t}} x^i_{s,t}. \end{aligned} \tag{13}$$

From (5) and (13), the entry of the Hessian for Q is

$$\frac{\partial^2 Q}{\partial \lambda_{u,v} \partial \lambda_{u',v'}}(q) = \sum_{s,t,i} D(s,t) K^{(u,v)}_{P^i_{s,t}} K^{(u',v')}_{P^i_{s,t}} x^i_{s,t}(q). \tag{14}$$

Since there are an infinite number of $x^i_{s,t}$, we need to find another way to compute the Hessian instead of directly using (14).

B. Framework of Solving NEM By Newton's Method

Algorithm 1 in [1] shows the framework of solving NEM by descent methods (including gradient descent and Newton's method). Starting with an initial setting of link weights, the algorithm repeatedly updates the link weights until the load on each link is the same as the necessary capacity. The *Traffic_Distribution* procedure computes the resulting link loads $f_{u,v}$, based on the traffic matrix and link weights. Then the *Link_Weight_Update* procedure updates link weights by a descent method.

```
Compute necessary capacities c̃ by solving (1)
w ← Any set of link weights
f ← Traffic_Distribution(w)
while f ≠ c̃ do
    w ← Link_Weight_Update(f)
    f ← Traffic_Distribution(w)
end while
Return w /*final link weights*/
```

Algorithm 1: Optimize Over Link Weights

Algorithm 2 shows the procedure of updating link weights by Newton's method. The methods of computing the link load $\boldsymbol{f}$ and Hessian $\boldsymbol{H}$ are shown in Sec. III-C and III-D. The time complexity mainly comes from computing the inverse of the $E \times E$ matrix H, which requires $O(E^3)$ time. In this paper, we only discuss the conceptual method of applying Newton's method. There exist several other faster methods of computing the inverse of the Hessian approximately via conjugate gradient, quasi-Newton methods, etc. [9].

```
for all links do
    w = w − H^{-1}(c̃ − f)
end for
```

Algorithm 2: Link-Weight_Update($\boldsymbol{f}$) by Newton's Method

C. Computing Link Load and Hessian By Exact PEFT

Equations (7) and (9) are called *Exact* PEFT, which corresponds to the exact solution of NEM [1]. Via (9), the Hessian can be computed without enumerating all the paths. At first, we define $\eta^{s,t}_u$ as the total flow at node u if there is one unit of traffic demand from s to t in the network where $s \neq t$. $\eta^{s,t}_u$ and f^t_u can be computed in the same way by solving linear equations (15), where (15a) is the same as (11).

$$f^t_s - \sum_{y:(y,s)\in\mathbb{E}} \psi^t_{y,s} f^t_y = D(s,t) \tag{15a}$$

$$\eta^{s,t}_u - \sum_{y:(y,u)\in\mathbb{E}} \psi^t_{y,u} \eta^{s,t}_y = \begin{cases} 1 & \text{if } u = s \\ 0 & \text{otherwise.} \end{cases} \tag{15b}$$

Therefore, for each destination t, (15) are linear equations $A\boldsymbol{x} = B$; A is $N \times N$, $\boldsymbol{x}$ is $N \times 1$, and B is $N \times N$. The total time complexity is $O(N \cdot N^3) = O(N^4)$ [10]. Note that solving (15a) alone also needs $O(N^4)$ time due to the LU Factorization of A for N destinations.

The Hessian can be computed using (16) from Theorem 1. It requires additional time $O(NE^2)$. Therefore, the total time of computing the Hessian is $O(N^4 + NE^2)$.

$$H_{(u,v)(u',v')} = \tag{16a}$$

$$= \begin{cases} \sum_{t\in\mathbb{V}} \left(f^t_{u,v} \eta^{v,t}_{u'} \psi^t_{u',v'} + f^t_{u',v'} \eta^{v',t}_u \psi^t_{u,v} \right) & \text{if } (u,v) \neq (u',v') \\ \sum_{t\in\mathbb{V}} \left(f^t_{u,v} (1 + 2\eta^{v,t}_u \psi^t_{u,v}) \right) & \text{if } (u,v) = (u',v'). \end{cases} \tag{16b}$$

***Theorem** 1:* The Hessian matrix for NEM dual (14) can be computed by (16).

Proof: See Appendix. ∎

D. Approximate Link Load and Hessian By Downward PEFT

Since the optimal traffic distribution should contain no cycles, the *Exact* PEFT (9) can be approximated by Downward PEFT (17) to forward traffic only on next hops which are closer to the destination [1].

$$\psi^t_{u,v} = \begin{cases} \frac{e^{-h^t_{u,v}}\Upsilon^t_v}{\Upsilon^t_u} & \text{if } d^t_u > d^t_v, \\ 0 & \text{otherwise.} \end{cases} \quad (17)$$

Then for each destination t, $\eta^{s,t}_u$ and f^t_u can be computed by visiting the nodes in decreasing topological order (from the farthest node to t) in an acyclic network [1]. The total time complexity is $N^2(N+E)$. Then the Hessian still can be computed using (16) but one of $\eta^{v,t}_{u'}\psi^t_{u',v'}$ and $\eta^{v',t}_u\psi^t_{u,v}$ should be 0 for a pair of links (u,v) and (u',v') in an acyclic network. In addition, $H_{(u,v)(u,v)} = \sum_{t\in\mathbb{V}} f^t_{u,v}$ since $\eta^{v,t}_u\psi^t_{u,v}$ is 0.

IV. Performance Evaluation

In this section, we numerically compare the performance of solving NEM by gradient descent algorithms and Newton's methods.

A. Simulation Environment

We use the same test setting as in [1]. We run the simulation for a real backbone network and several synthetic networks. First is the Abilene network with 11 nodes and 28 directional links with 10Gbps capacity. The traffic matrix is derived from the Netflow data on Nov. 15th, 2005. The two synthetic 2-level hierarchical networks were generated using GT-ITM, which consists of two kinds of links: local access links with 200-unit capacity and long distance link with 1000-unit capacity. In the two random topologies, the probability of having a link between two nodes is a constant parameter and all link capacities are 1000 units [7].

As in [1], we adopt the cost function (18) in [7] and the objective is to minimize $\sum_{(u,v)} \Phi(f_{u,v}, c_{u,v})$.

$$\Phi(f_{u,v}, c_{u,v}) = \begin{cases} f_{u,v} & f_{u,v}/c_{u,v} \le 1/3 \\ 3f_{u,v} - 2/3\, c_{u,v} & 1/3 \le f_{u,v}/c_{u,v} \le 2/3 \\ 10f_{u,v} - 16/3\, c_{u,v} & 2/3 \le f_{u,v}/c_{u,v} \le 9/10 \\ 70f_{u,v} - 178/3\, c_{u,v} & 9/10 \le f_{u,v}/c_{u,v} \le 1 \\ 500f_{u,v} - 1468/3\, c_{u,v} & 1 \le f_{u,v}/c_{u,v} \le 11/10 \\ 5000f_{u,v} - 16318/3\, c_{u,v} & 11/10 \le f_{u,v}/c_{u,v}, \end{cases} \quad (18)$$

The optimal Φ values are computed by solving linear program (1) with CPLEX 9.1 via AMPL. For each network, we uniformly scale the traffic matrix such that the maximum link utilization is close to 100% in the optimal solution. Note that such optimal Φ value is not the maximum entropy in NEM, which is not available beforehand. However, the network operators mainly care about the Φ value and from the theory of NEM [1], the descent methods will converge to a set of link weights which can *simultaneously* achieve the minimum Φ value and the maximum entropy using PEFT. Then it is still meaningful to show the convergence of optimality gap, in terms of the Φ value, compared against the Φ value achieved by optimal traffic engineering (1).

To solve NEM, i.e., to determine link weights under PEFT, we run Algorithm 1 with up to 5000 iterations for gradient descent and 500 iterations for Newton's method of computing traffic distribution and updating link weights for the two scenarios: Exact PEFT (with cycles) and Downward PEFT (without cycles). Then we have four algorithms named Exact-Grad, Down-Grad, Exact-Newton and Down-Newton. Each algorithm will terminate earlier if the optimality gap is already less than 1%. In gradient descent algorithms, the step size is set to $1/\max \widetilde{c}_{u,v}$, the reciprocal of the maximum necessary link capacity [1], while the step size of Newton's method is always 1. To determine the flow with cycles and compute the inverse of the Hessian, we solve linear equations via LAPACK (Linear Algebra PACKage) [11].

B. Convergence Behavior

Fig. 1 shows the optimality gap (on a log scale) achieved by PEFT of different algorithms, within the first 3000 iterations for six networks. It demonstrates that Downward PEFT is indeed a good approximation of Exact PEFT though the former requires a few more iterations. In addition, solving NEM by Newton's methods converges much faster than by gradient descent. For example, in the Abilene network, Newton's method only needs 43 iterations to reduce the optimality gap to 1%, while gradient descent needs 2955 (or 68 times as many) iterations. Similar observations can be made for other networks in Fig. 1.

C. Running Time Requirement

The tests for finding link weights in PEFT with different algorithms were performed under a time-sharing server of Centos Linux 4.4 with Intel Xeon processors at 2.6 Ghz. Table II shows the average running time per iteration of various algorithms on six networks. We observe that all the four algorithms are very fast, requiring at most 7 minutes before termination even for the largest network (with 100 nodes) tested. Moreover, approximating Exact PEFT by Downward PEFT can reduce the running time by half or even more due to the avoidance of solving linear equations to determine the flow with cycles. In each iteration, Newton's method requires more time than needed by gradient descent algorithms because of the complexity of calculating the inverse of the Hessian, which fortunately can be considerably simplified by adopting some quasi-Newton methods [12] (which we don't do).

TABLE II
Average running time per iteration required by solving NEM using gradient descent and Newton's methods

Net. ID	Node #	Link #	Time per Iteration (millisecond)			
			Gradient		Newton	
			Exact	Down	Exact	Down
abilene	11	28	3.4	2.1	7.9	3.6
hier50a	50	148	13.7	5.9	57.1	17.7
hier50b	50	212	13.8	6.3	82.4	36.4
rand50	50	228	19.4	7.1	94.8	43.9
rand50a	50	245	24.3	7.6	110.8	50.0
rand100	100	403	175.5	36.7	911.7	282.3

V. Conclusion

Optimal traffic engineering can be realized by a link-state routing protocol by solving a convex optimization problem, NEM. In this paper, we propose efficient Newton's methods to

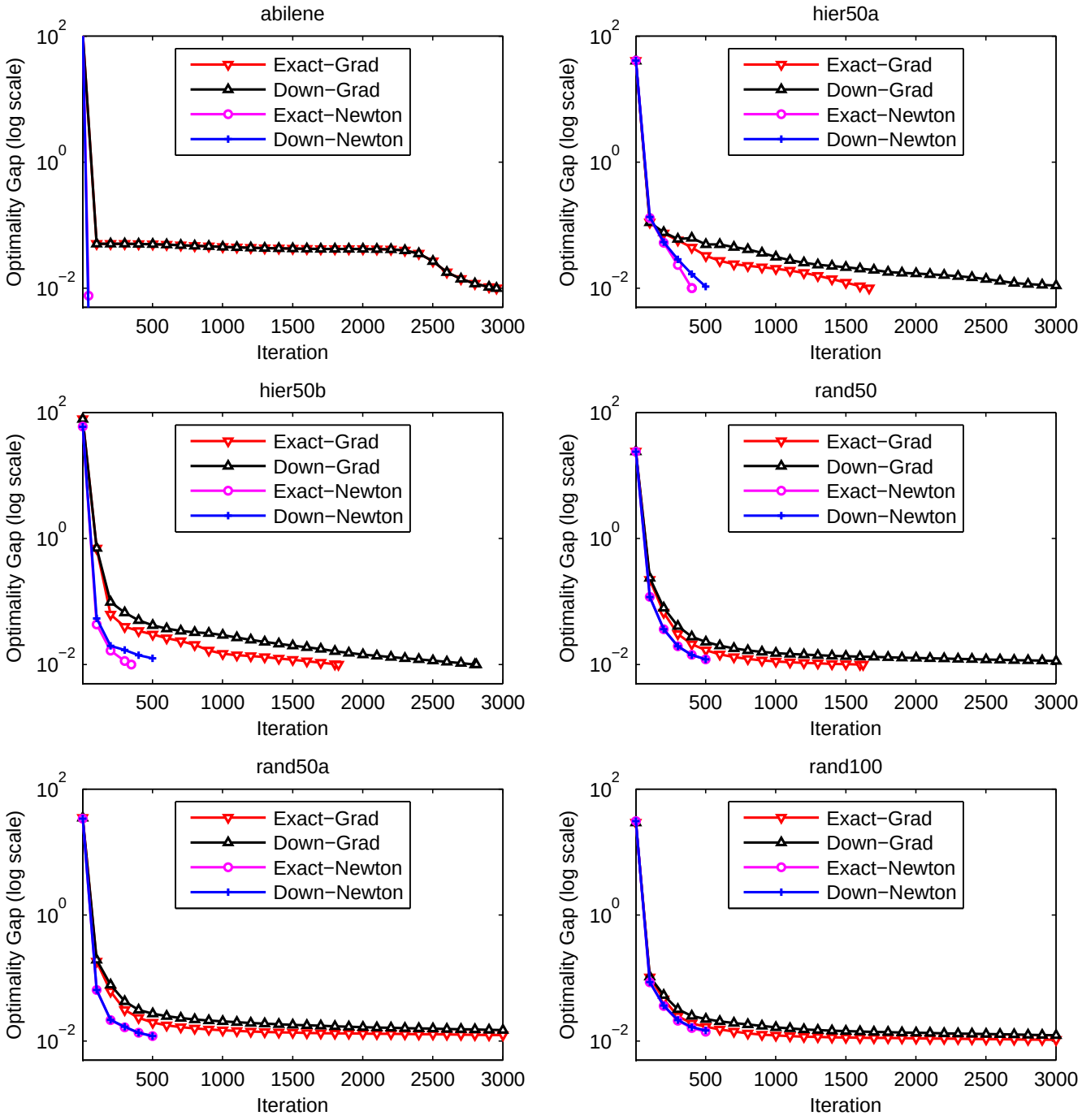

Fig. 1. Evolution of optimality gap using Exact and Downward PEFT with gradient descent and Newton's methods

solve the NEM problem with an infinite number of variables. The extensive simulation validates that the proposed Newton method converges much faster than the existing gradient descent algorithms for many practical instances in addition to converging faster in theory.

ACKNOWLEDGMENT

We would to thank Howard Karloff for the very valuable comments. We also appreciate the helpful discussions with Mung Chiang, Ying Li, Jennifer Rexford, and Chao Tian.

APPENDIX: PROOF OF THEOREM 1

Proof: Denote by $\xi_u^{t,(u_1,v_1)}$ the expected number of passages through link (u_1, v_1) for a packet traveling from u to t and let $\xi_t^{t,(u_1,v_1)} = 0$. Define function $\delta(\text{condition})$ to be 1 if the condition is true, and 0 otherwise. Then $\xi_u^{t,(u_1,v_1)}$ can be computed using the following linear equations (19).

$$\xi_u^{t,(u_1,v_1)} \triangleq \sum_{v:(u,v)\in\mathbb{E}} \xi_v^{t,(u_1,v_1)} \psi_{u,v}^t + \psi_{u_1,v_1}^t \cdot \delta(u = u_1). \quad (19)$$

From the definition, $\xi_u^{t,(u_1,v_1)}$ also can be computed by solving linear equations (15) and applying (20).

$$\xi_u^{t,(u_1,v_1)} = \eta_{u_1}^{u,t} \psi_{u_1,v_1}^t \quad (20)$$

Similarly, denote by $\xi_u^{t,(u_1,v_1),(u_2,v_2)}$ the expected *product* of the number of passages through link (u_1, v_1) and the number of passages through link (u_2, v_2); let $\xi_t^{t,(u_1,v_1),(u_2,v_2)} = 0$. If $(u_1, v_1) = (u_2, v_2)$, then $\xi_u^{t,(u_1,v_1),(u_2,v_2)}$ is the expected *square* of the number of passages through link (u_1, v_1). $\xi_u^{t,(u_1,v_1),(u_2,v_2)}$ can be computed by solving linear equations (21).

$$\xi_u^{t,(u_1,v_1),(u_2,v_2)} = \sum_{v:(u,v)\in\mathbb{E}} \xi_v^{t,(u_1,v_1),(u_2,v_2)} \psi_{u,v}^t + \begin{cases} \psi_{u_1,v_1}^t \xi_{v_1}^{t,(u_2,v_2)} \cdot \delta(u = u_1) + \psi_{u_2,v_2}^t \xi_{v_2}^{t,(u_1,v_1)} \cdot \delta(u = u_2) & \text{if } (u_1, v_1) \neq (u_2, v_2) \\ \psi_{u_1,v_1}^t (1 + 2\xi_{v_1}^{t,(u_1,v_1)}) \cdot \delta(u = u_1) & \text{if } (u_1, v_1) = (u_2, v_2) \end{cases} \quad (21)$$

From (19) and (20), the solution for (21) is shown in (22).

$$\xi_u^{t,(u_1,v_1),(u_2,v_2)} = \begin{cases} \eta_{u_1}^{u,t} \psi_{u_1,v_1}^t \eta_{u_2}^{v_1,t} \psi_{u_2,v_2}^t + \eta_{u_2}^{u,t} \psi_{u_2,v_2}^t \eta_{u_1}^{v_2,t} \psi_{u_1,v_1}^t & \text{if } (u_1, v_1) \neq (u_2, v_2) \\ \eta_{u_1}^{u,t} \psi_{u_1,v_1}^t (1 + 2\eta_{u_1}^{v_1,t} \psi_{u_1,v_1}^t) & \text{if } (u_1, v_1) = (u_2, v_2) \end{cases} \quad (22)$$

Then

$$f_{u_1,v_1}^t = \sum_s D(s,t) \xi_s^{t,(u_1,v_1)} = \sum_s D(s,t) \eta_{u_1}^{s,t} \psi_{u_1,v_1}^t \quad (23)$$

and with (14), (22) and (23), we have

$$H_{(u_1,v_1),(u_2,v_2)} = \sum_{s,t} D(s,t) \xi_s^{t,(u_1,v_1),(u_2,v_2)} = \begin{cases} f_{u_1,v_1}^t \eta_{u_2}^{v_1,t} \psi_{u_2,v_2}^t + f_{u_2,v_2}^t \eta_{u_1}^{v_2,t} \psi_{u_1,v_1}^t & \text{if } (u_1, v_1) \neq (u_2, v_2) \\ f_{u_1,v_1}^t (1 + 2\eta_{u_1}^{v_1,t} \psi_{u_1,v_1}^t) & \text{if } (u_1, v_1) = (u_2, v_2). \end{cases} \quad (24)$$

■

REFERENCES

[1] D. Xu, M. Chiang, and J. Rexford, "Link-state routing with hop-by-hop forwarding can achieve optimal traffic engineering," in *INFOCOM'08, Phoenix, AZ*, Apr. 2008.
[2] Z. Wang, Y. Wang, and L. Zhang, "Internet traffic engineering without full mesh overlaying," in *INFOCOM'01, Anchorage, AK*, 2001.
[3] L. Buriol, M. Resende, C. Ribeiro, and M. Thorup, "A memetic algorithm for OSPF routing," in *Proceedings of the 6th INFORMS Telecom*, 2002, pp. 187–188.
[4] A. Sridharan, R. Guérin, and C. Diot, "Achieving near-optimal traffic engineering solutions for current OSPF/IS-IS networks," *IEEE/ACM Transactions on Networking*, vol. 13, no. 2, pp. 234–247, 2005.
[5] J. H. Fong, A. C. Gilbert, S. Kannan, and M. J. Strauss, "Better alternatives to OSPF routing," *Algorithmica*, vol. 43, no. 1-2, pp. 113–131, 2005.
[6] D. Xu, M. Chiang, and J. Rexford, "DEFT: Distributed exponentially-weighted flow splitting," in *INFOCOM'07, Anchorage, AK*, May 2007.
[7] B. Fortz and M. Thorup, "Internet traffic engineering by optimizing OSPF weights," in *INFOCOM'00, Tel Aviv, Israel*, 2000, pp. 519–528.
[8] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Athena Scientific, 1999.
[9] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.
[10] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*. The MIT Press, Cambridge, 1990.
[11] LAPACK, http://www.netlib.org/lapack/.
[12] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.