Faster combinatorial approximation algorithms for multi-commodity flow problems and its application[1]

Suh-Wen Chiou

Department of Information Management, National Dong Hwa University

1, Sec. 2, Da Hsueh Rd., Shou-Feng, Hualien, 974, Taiwan

Abstract

A multi-commodity flow problem is to find a feasible flow that satisfies the demand of each source to the sink in a directed network while the flow on each edge is within its capacity. In this paper we implement polynomial-time combinatorial approximation algorithms for ε-optimal multi-commodity flow problems. A new algorithm Optima is proposed and good test results are obtained. Application of Optima to a system-optimized multi-user network flow problem is given where substantially good test results have shown the capacity of Optima in dealing with problems of multi-commodity flow associated.

Keywords： combinatorial optimization algorithms, minimum-cost flows, multi-commodity flow.

[1] Paper submitted to 2002 International Computer Symposium (ICS 2002) Workshop on Algorithms and Computational Molecular Biology

1. Introduction

A multi-commodity flow problem is to find a feasible flow that satisfies the demand of each source to the corresponding sink in a directed network while the flow on each edge is within its capacity. The concurrent flow problem is an optimized version of the multi-commodity flow problem, where the objective is to find the maximal value of the fraction z such that the least z percent of each demand can be assigned while for every edge the capacity constraint is satisfied. Throughout this paper, we use n, m and k to denote the number of nodes, edges and commodities.

Leighton, Makedon, Plotkin, Stein, Tardos and Tragoudas [3] prpposed a polynomial-time combinatorial approximation algorithm for the concurrent flow problem. In Leighton's paper, given the precision value ε, the running time of the deterministic version is in $O(k^2\varepsilon^{-2}\log(\frac{n}{\varepsilon}) + k^2 \log n \log k)$ minimum-cost flow computations and the running time of the randomized version is in $O(k\varepsilon^{-3}\log(\frac{n}{\varepsilon}) + k \log n \log k)$ minimum-cost flow computations. Goldberg [2] gave a natural randomization strategy and simplified in $O(k\varepsilon^{-2}\log(\frac{n}{\varepsilon}) + k \log n \log k)$ minimum-cost flow computations by a fraction of ε^{-1}. Radzik [7] proposed a fast deterministic approximation algorithm and revised the deterministic version of Leighton by $O(k\varepsilon^{-2}\log(\frac{n}{\varepsilon}) + k \log n \log k)$ minimum-cost flow computations, where a tighter bound for the multi-commodity flow approximation can be computed deterministically.

Leong, Shor and Stein [4] implemented Leighton's algorithm on a medium sized

network to solve the multi-commodity flow problem. Comparisons of the effectiveness for Leighton's method over the other two conventional algorithms: linear programming and interior point are reported. In this paper, following the work of Leighton and Radzik, we implement the two best known polynomial-time combinatorial approximation methods on a large scale test networks. An optimization based method for searching the optimal fraction of each commodity demand is proposed and good test results are also given. Application of the optimized method to the problem of transport studies (Chiou, [1]) is presented. The remaining of this paper is organized as follows. In Section 2, the preliminaries and the definitions for multi-commodity flow problem are given. Algorithms of Leighton's (Decongest) and of Radzik (Improve) are respectively described in Section 3. In Section 4, an optimization based search method (Optima) in determining the optimal fraction of each commodity demand is proposed. In Section 5, implementations of the best two known combinatorial approximation algorithms for multi-commodity flow problems are made. Also test results for our optimized search method-Optima are given. A system-optimized multi-user network flow is illustrated when Optima applies to the problem of transport studies. In Section 6, conclusion and discussions for solving the multi-commodity flow problem and its application are made.

2. Preliminaries and Definitions

Let $G = (V, E)$ denote a directed network, where V and E respectively represent the set of nodes and edges. An input instance is the capacity function for each edge $u : V \to \Re^+$. For the specification of each commodity $i, 1 \le i \le k$, it contains the source s_i, the destination t_i and the nonnegative demand d_i. For each commodity flow f_i, where $f_i : E \to \Re^+$, the flow conservation on each edge must be satisfied. For each

node v in V,

$$\sum_{(u,v)\in E} f_i(u,v) - \sum_{(v,u)\in E} f_i(v,u) = \begin{cases} d_i & v = t_i \\ -d_i & v = s_i \\ 0 & otherwise \end{cases} \quad (1)$$

Let $f(e) = \sum_{1\le i\le k} f_i(e), \forall e \in E$, the capacity constraint must be satisfied: $\forall e \in E, f(e) \le u(e)$. Since the concurrent flow problem is an optimized version of the multi-commodity flow problem for which the fraction z is maximized such that the flow with zd_i, is feasible, in other words, the dual problem for the concurrent flow problem is to find the minimum congestion λ such that the flow with d_i, is feasible when the capacity is $\lambda u(e), \forall e \in E$. Let $\lambda(e) = \frac{f(e)}{u(e)}, \lambda = \underset{\forall e\in E}{Max}\lambda(e)$. λ^* is the optimal value of λ, in this paper, we focus on the ε-optimal solution to the concurrent flow problem, where a multi-commodity flow is ε-optimal if $\lambda \le (1+\varepsilon)\lambda^*$. Given a parameter α, a length function $l: E \to \Re^+$ can be defined as an exponential function form of congestion (Shahrokhi and Matula [9])

$$l(e) = \exp(\alpha f(e)), \forall e \in E \quad (2)$$

An objective function therefore can be defined as

$$\Phi = \sum_{e\in E} l(e)u(e) \quad (3)$$

With respect to Leighton *et al.* and Rdazik, the following mathematical conditions are given straightforward without proof.

Lemma 1. Let C_i be the cost of concurrent flow for commodity i, where

$C_i = \sum_{\forall e \in E} f_i(e) l(e)$ and $C_i^*(\lambda)$ be the value of minimum-cost flow with congestion λ, then

$$\sum_{1 \le i \le k} C_i^*(\lambda) \le \sum_{1 \le i \le k} C_i \le \lambda \Phi \tag{4}$$

Lemma 2. For any length function l, there exists a multi-commodity flow satisfying capacity $\lambda u(e), \forall e \in E$ such that

$$\sum_{1 \le i \le k} C_i^*(\lambda) \le \lambda^* \Phi \tag{5}$$

Theorem 3. (relaxed optimality conditions, Radzik) For $\lambda \ge \lambda^*$ given a precision $\varepsilon \le \frac{1}{3}$, for a multi-commodity flow f satisfying capacity $\lambda u(e), \forall e \in E$ such that for every edge e

$$\lambda(e) \le \left(1 + \frac{\varepsilon}{3}\right) \lambda \tag{6}$$

and

$$\left(1 - \frac{\varepsilon}{2}\right) \lambda \Phi \le \sum_{1 \le i \le k} C_i^*(\lambda) \tag{7}$$

then multi-commodity flow f is ε-optimal.

3. Polynomial-time combinatorial algorithms

Two best known polynomial-time combinatorial algorithms for ε-optimal concurrent flow problem are given. Decongest was proposed by Leighton *et al.* and Improve was presented by Radzik after modified by Goldberg.

3.1 Decongest

For a given multi-commodity flow f with congestion λ_0 and a precision value ε, let parameter $\alpha = \frac{2(1+\varepsilon)}{\lambda_0 \varepsilon} \ln\left(\frac{m}{\varepsilon}\right)$. At each iteration a 'bad' commodity flow f_i was chosen such that

$$C_i - C_i^*(\lambda) > \varepsilon C_i + \frac{\varepsilon\lambda}{k}\Phi \tag{8}$$

Build an auxiliary minimum-cost flow problem and find a minimum-cost flow f_i^* which minimizes $C_i^*(\lambda)$ in the auxiliary network. Use a given fraction value $\sigma, \sigma = \frac{\varepsilon}{8\alpha\lambda}$, compute the updated flow for each edge e by $f_i(e) \leftarrow \sigma f_i^*(e) + (1-\sigma) f_i(e)$. Repeat the forgoing step until the congestion $\lambda < \frac{\lambda_0}{2}$ or $(1+9\varepsilon)\lambda \geq \lambda^*$.

3.2 Improve

Given an initial current flow and a precision value ε, the purpose of this algorithm is to improve the concurrent 'bad' commodity flows and reduce the gap between the commodity flow cost and the one in the minimum-cost flows. Radzik presented similar steps as Decongest, which is stated below. Let $f = (f_1, f_2, ..., f_k)$ be the concurrent flow with congestion λ_0 at the beginning of one iteration of the outer loop. For iteration j, within capacities $\lambda_0 u\left(1+\frac{\varepsilon}{3}\right)$ and with respect to current length function, we compute a minimum-cost flow f_i^* of commodity flow f_i. If commodity flow f_i is a 'bad' flow, that is,

$$C_i - C_i^*(\lambda) \geq \varepsilon C_i \tag{9}$$

A strategy to improve current 'bad flow' of commodity i can be given as

$$f_i^{'} = f_i + \sigma\left(f_i^* - f_i\right) \tag{10}$$

where $\sigma = \frac{\varepsilon}{4\alpha\lambda}$ and $\alpha = \frac{3(1+\varepsilon)}{\lambda_0 \varepsilon}\ln\left(\frac{m}{\varepsilon}\right)$.

Thus at iteration j the current flow becomes $f^{(j)} = \left(f_1^{'},...,f_i^{'},f_{i+1},...f_k\right)$. Update the new length function $l^{(j)} = l(f^{(j)})$ and the corresponding objective values $\Phi^{(j)} = \Phi\left(f^{(j)}\right)$, then compute the next 'bad' commodity flow. Continue such computation for each commodity for iterations until the congestion value is not greater than $\left(1-\frac{\varepsilon}{3}\right)\lambda_0$ or the objective value Φ fails to decrease by a factor of $\left(1-\frac{\varepsilon^2}{8}\right)$. Then an ε optimal flow can be computed by $O(k\varepsilon^{-2}\log(\frac{n}{\varepsilon}) + k\log n\log k)$ minimum-cost flow computations.

4. Optimal search for ε -Optima

Following the algorithms of Decongest and Improve, we can find an approximation concurrent flow. Given a precision value ε, the polynomial-time computations are guaranteed by $O\left(k\log(k)\log(n)\right)$ when the number of commodities increase. However, in relation to the way in deciding the optimal fraction of the convex combination for minimum-cost flow and the 'bad' commodity flow, a given formula without ant optimal analysis was used for both of the two algorithms. In this section, a new algorithm Optima is proposed below by employing the bisection method to find

the optimal value of the fraction σ of each commodity flow f_i without increasing current computations. The pseudo code for Optima is given in Table 1.

For a given concurrent flow f and a precision value ε, compute the initial congestion λ_0. Let $\alpha = \frac{1}{\lambda_0 \varepsilon} \ln\left(\frac{m}{\varepsilon}\right)$, compute the initial value of the objective function Φ. For each commodity flow f_i, find the minimum-cost flow f_i^* and check if this commodity flow is a 'bad commodity flow', e.g. $C_i - C_i^*(\lambda) \geq \varepsilon C_i$. If so, do Optima-$\sigma$, and the commodity flow can be updated as

$$f_i \leftarrow f_i + \sigma(f_i^* - f_i) \tag{11}$$

Termination conditions are used as the same as given in Improve.

The Optima-σ search procedure is conducted as follows.

Let $[a_1, b_1]$ be the initial interval of uncertainty, and let γ be the allowable final interval of uncertainty. Let ω be the smallest positive integer such that

$$\left(\frac{1}{2}\right)^{\omega} \leq \frac{\gamma}{b_1 - a_1} \tag{12}$$

At iteration j:

Step 1. Let $\sigma_j = \frac{a_j + b_j}{2}$ and evaluate the first derivative, $\frac{d\Phi(\sigma_j)}{d\sigma_j}$, of Φ. If $\frac{d\Phi(\sigma_j)}{d\sigma_j} = 0$, stop and σ_j is optimal. Otherwise, go to step 2 if $\frac{d\Phi(\sigma_j)}{d\sigma_j} > 0$, and go to step 3 if $\frac{d\Phi(\sigma_j)}{d\sigma_j} < 0$.

Step 2. Let $a_{j+1} = a_j$ and $b_{j+1} = \sigma_j$, go to step 4.

Step 3. Let $a_{j+1} = \sigma_j$ and $b_{j+1} = b_j$, go to step 4.

Step 5. If $j = \omega$, stop. The optimal value of σ for the minimum objective function Φ with respect to commodity flow f_i is within the interval $[a_{\omega+1}, b_{\omega+1}]$. Otherwise, increase j by 1 and repeat step 1.

5. Implementations and application

In this section, algorithms of Decongest, Improve, Improve-variant and Optima were tested for theoretical bound examination and implementation efficiency. Application of Optima to a problem of system-optimized multi-user network flow in transport studies was illustrated. Implementations conducted below were made on Sun SPARC ultra 30 with the operating system Unix SunOS 5.5.1. Computer programs were coded in C++ language and complied with GNU g++ 2.8.1.

5.1 Implementations

Algorithms of Decongest, Improve and the Improve-variant were implemented on the test networks which were produced by 'NetworkGen'. 'NettworkGen' [6] is a network generator which has been developed under the environment provided by LEDA (Mehlhorn, Naher and Uhrig [5]) and written in c++ computer language. In this subsection, we firstly examined the polynomial-time bound for the increase in the number of commodities and the ε value. We extended the algorithm Improve to Improve-variant by giving the value of σ as $\frac{\varepsilon^2}{\log n}$ as proposed by Radzik [8]. In figures 1-3, we tested algorithms of Decongest, Improve and Improve-variant at the

size of networks denoted by $n/m/k$, which is short for $n = 50, m = 250$, and for the variety of number of commodities $k = 10,40,50,70,90,100,120,150,180,200$. As it seen from figures 1-3, the computation times for the increase in the number of commodities were clearly below the polynomial-time bound, which agreed with theoretical prediction of the polynomial-time bound in the number of commodities. Given a variant of the precision value, e.g. $\varepsilon \in [0.01,0.1]$, results shown in figures 4-6 have been strongly confirmed to the theoretical bound of $O(\varepsilon^{-2})$.

Furthermore, following earlier discussions for Optima, we tested the four algorithms of Decongest, Improve, Improve-variant and Optima on large scale networks of 100/1000/200 and 200/600/500, for computing the efficiency in decreasing the value of congestion and the objective function in terms of the number of computations achieved. As it seen in figures 7-10, algorithm Optima gave substantially good results as compared to Decogest, Improve and Improve-variant for ε-optimal convergence.

5.2 Application

A problem of system-optimized multi-user network flow is to find an optimal network flow where a given objective function in terms of the sum of total users' travel costs is minimized. Following the work of Chiou, we applied the algorithm of Optima to solve the system-optimized network flow problem. Good results have been shown in figures11-12 for the values of congestion and the objective functions in a variety of networks 100/200/100, 100/200/120, 100/200/140 and 100/200/160. As found in figures 11-12, the number of computations for each run are less than 50 and the average of each running time is less than 0.1 second cpu time.

6. Conclusion and discussions

In this paper, we implemented Leighton's Decongest, Radzik's Improve and the extension of Improve-variant on a large scale of test networks. As expected, the examined results are in agreement with theoretical prediction in polynomial-time bound for the number of commodities and the precision values. In order to decide the optimal value of σ, we proposed algorithm Optima and tested its computing efficiency by comparing the two best known combinatorial algorithms.

Good results have been obtained. Also application of algorithm Optima to a problem of system-optimized multi-user network flows has been done. Again, encouraging results have been shown the capacity of algorithm Optima in solving the application problems of multi-commodities in transport studies. Further work needs to be taken for exploring the computation complexity when apply to different problem by Optima.

7. Acknowledgements

We are grateful to Dr. Radzik for his discussions on the earlier version of this work when the author conducted her post-doctorships at Department of Computer Science, King' s College London, which granted by U.K. EPSRC. This work reported has been sponsored by Taiwan National Science Council via grant NSC 90-2416-H-036-001.

8. References

[1] S-W. Chiou, "Optimization of area traffic control for equilibrium network flows", Transportation Science, 33, pp. 279-289, 1999.

[2] A.V. Goldberg, "A natural randomization strategy for multicommodity flow and related algorithms", Information Processing Letters, 42, pp. 249-256, 1992.

[3] T. Leighton, F. Makedon, S. Plotkin, C. Tardos and S. Tragoudas," Fast approximation algorithms for multicommodity flow problems", Journal of Computer and System Sciences, 50, pp. 228-243, 1995.

[4] T. Leong, P. Shor and C. Stein, "Implementation of a combinatorial multicommodity flow algorithm" in DIMACS series in Discrete Mathematics and Theoretical Computer Science, Vol 12, (American Mathematical Society, Providence, RI, 1993), pp. 387-405.

[5] K. Melhorn, S. Naher and C. Uhrig, The LEDA User Manual, version 3.8. Max-Planc-Institut fur Informatik, 66213 Saarbrucken, Germany, 1999

[6] NetworkGen, software developed by Supply Chain Management Lab, Department of Information Management, Tatung University. 2000

[7] T. Radzik, "Fast deterministic approximation for the multicommodity flow problem", Mathematical Programming, 78, pp. 43-58, 1997.

[8] T. Radzik, Private communication, 1999.

[9] F. Shahrokhi and D.W. Matula, " The maximum concurrent flow problem", Journal of the Association for Computer Machinery, 37, pp. 318-334, 1990.

Table 1: Algorithm Optima

Given current flow $f = (f_1,..,f_k)$ and $\varepsilon > 0$

$\lambda_0 = \lambda$;

repeat outer iteration, e.g. for iteration j ;

$f^{(0)} = f, \Phi^{(0)} = \Phi$;

for each commodity i **do**

find minimum-cost flow f_i^* ;

check 'bad commodity flow', if so,

do Optima-σ ;

then $f_i \leftarrow f_i + \sigma(f_i^* - f_i)$

update $f^{(j)}, \Phi^{(j)}$;

$f \leftarrow f^{(j)}$;

until $\lambda \le \left(1 - \frac{\varepsilon}{3}\right)\lambda_0$ or $\frac{\Phi^{(0)} - \Phi^{(j)}}{\Phi^{(0)}} \le \frac{\varepsilon^2}{8}$;

<Optima-σ>

given the initial interval $[a_1, b_1]$ and let the threshold γ :

begin find ω such that $\left(\frac{1}{2}\right)^{\omega} \le \frac{\gamma}{b_1 - a_1}$;

while ($j < \omega$ or $\frac{d\Phi(\sigma_j)}{d\sigma_j} \ne 0$, where $\sigma_j = \frac{a_j + b_j}{2}$)

do update search interval according to the sign of $\frac{d\Phi(\sigma_j)}{d\sigma_j}$:

if $\frac{d\Phi(\sigma_j)}{d\sigma_j} > 0, [a_{j+1}, b_{j+1}] = [a_j, \sigma_j]$;

else if $\frac{d\Phi(\sigma_j)}{d\sigma_j} < 0, [a_{j+1}, b_{j+1}] = [\sigma_j, b_j]$;

repeat j ;

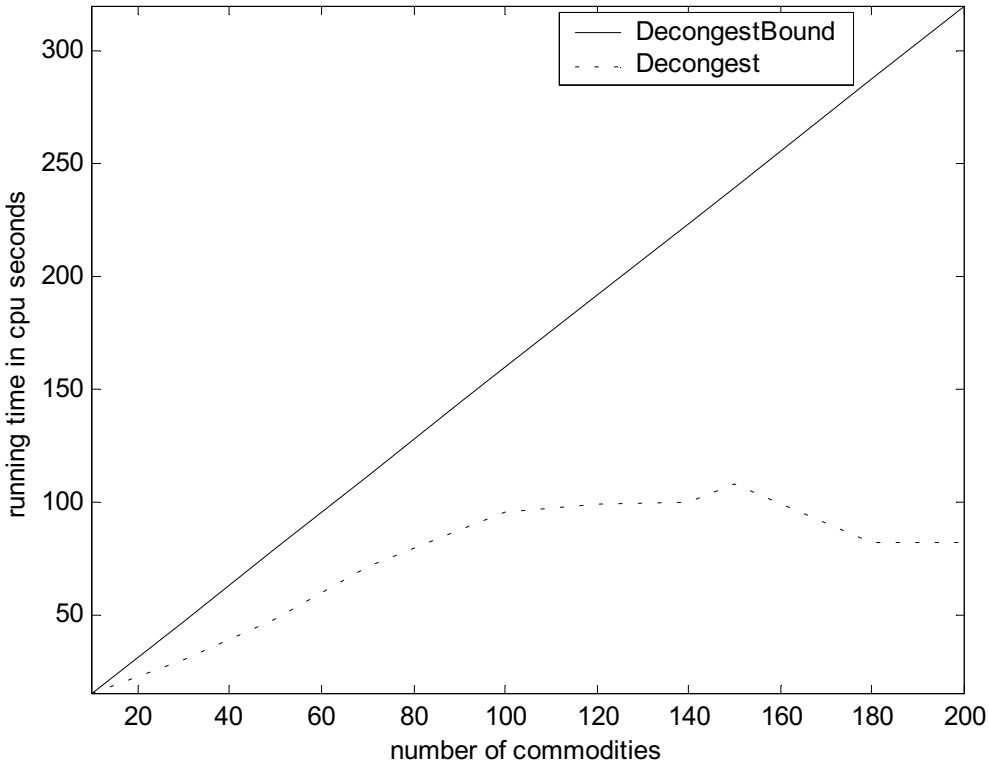

Figure 1. Polynomial-time bound for Decongest in n=50, m=250

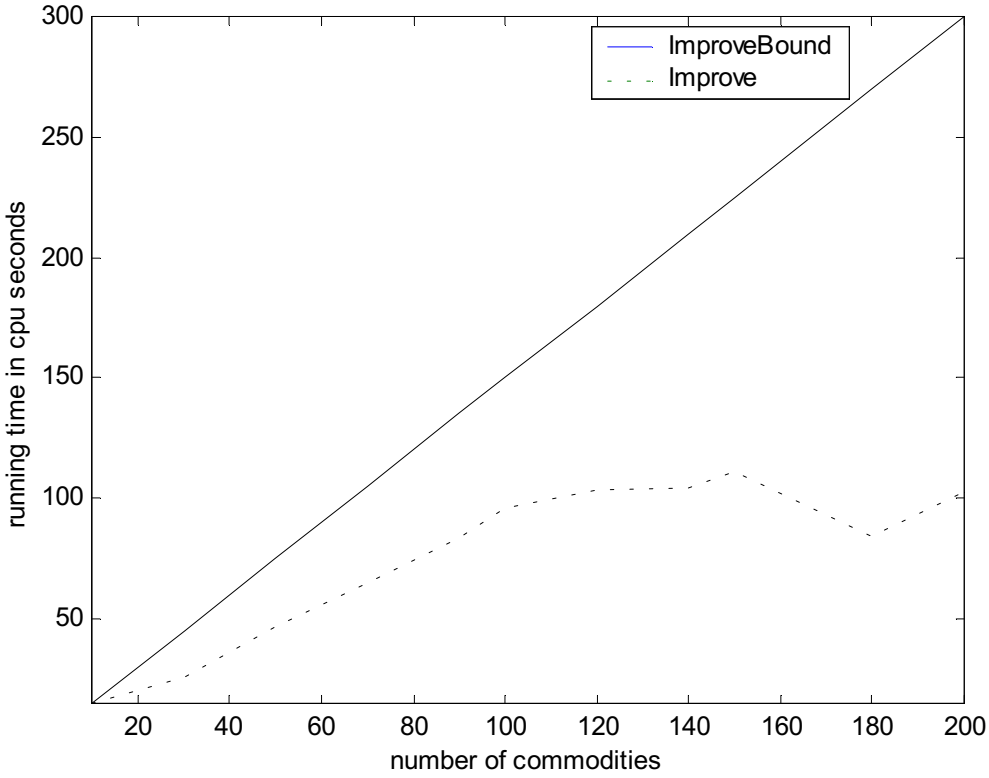

Figure 2. Polynomial-time bound for Improve in n=50, m=250

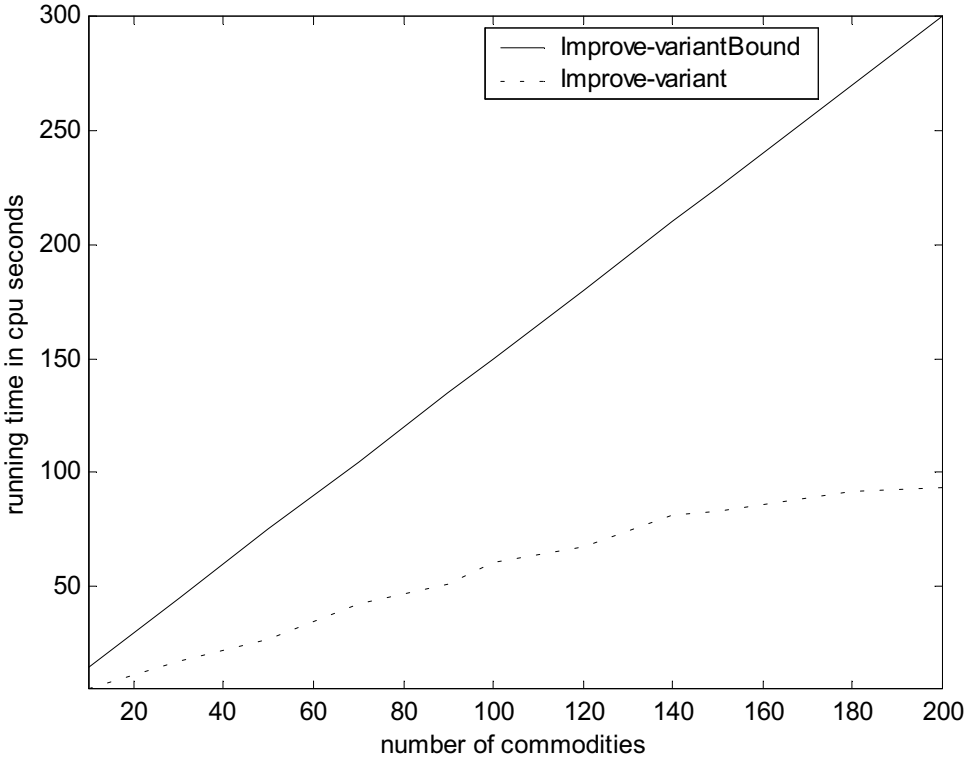

Figure 3. Polynomial-time bound for Improve-variant in n=50, m=250

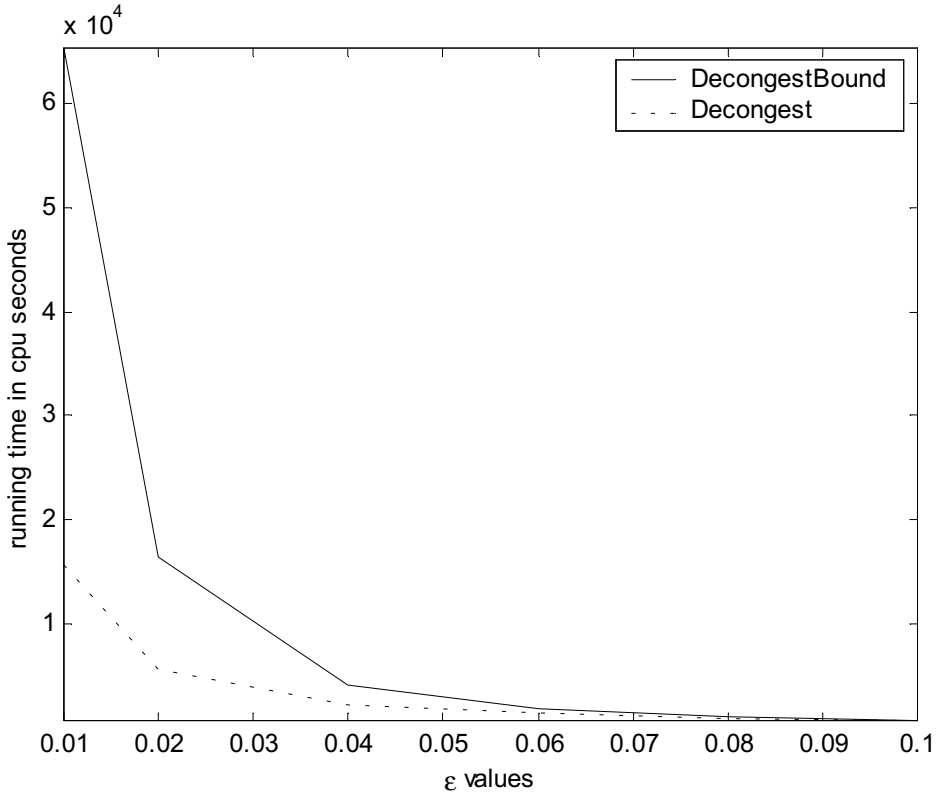

Figure 4. ε polynomial-time bound for Decongest, n=50, m=250, k=70

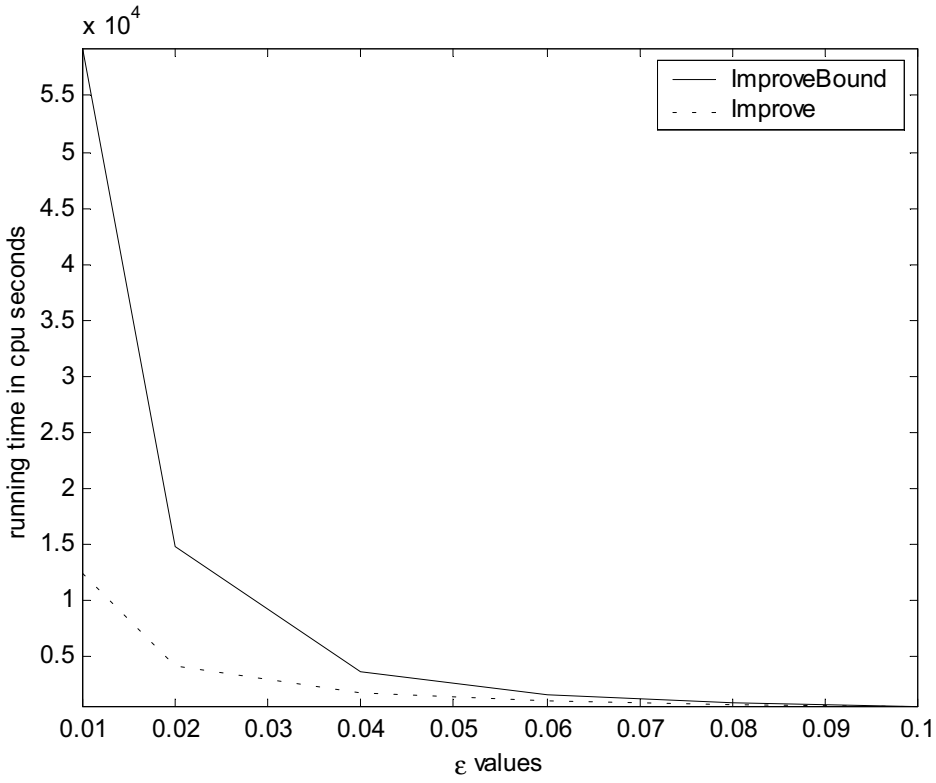

Figure 5. ε polynomial-time bound for Improve, n=50, m=250, k=70

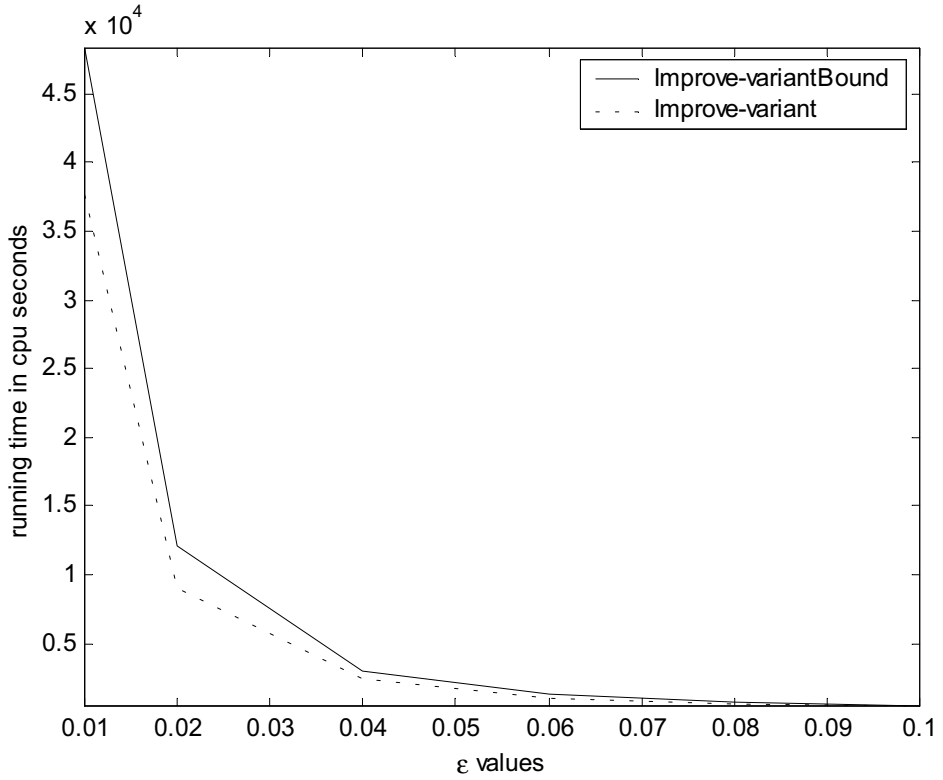

Figure 6. ε polynomial-time bound for Improve-variant, n=50, m=250, k=70

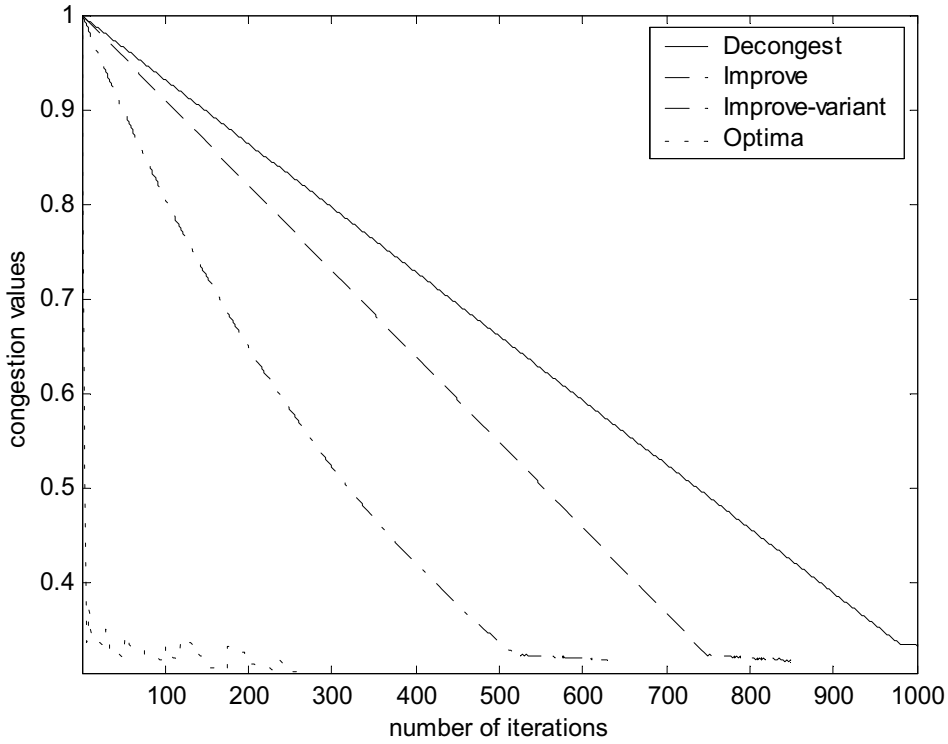

Figure 7. Congestion analysis for ε =0.1, n=100, m=1000, k=200

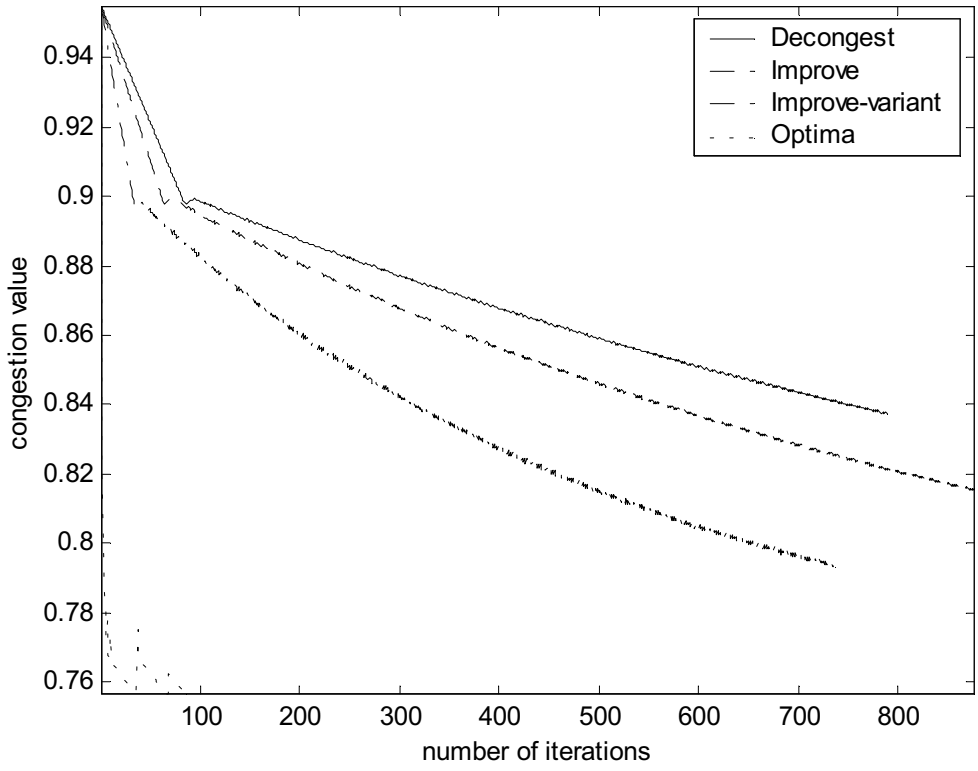

Figure 8. Congestion analysis for ε =0.1, n=200, m=600, k=500

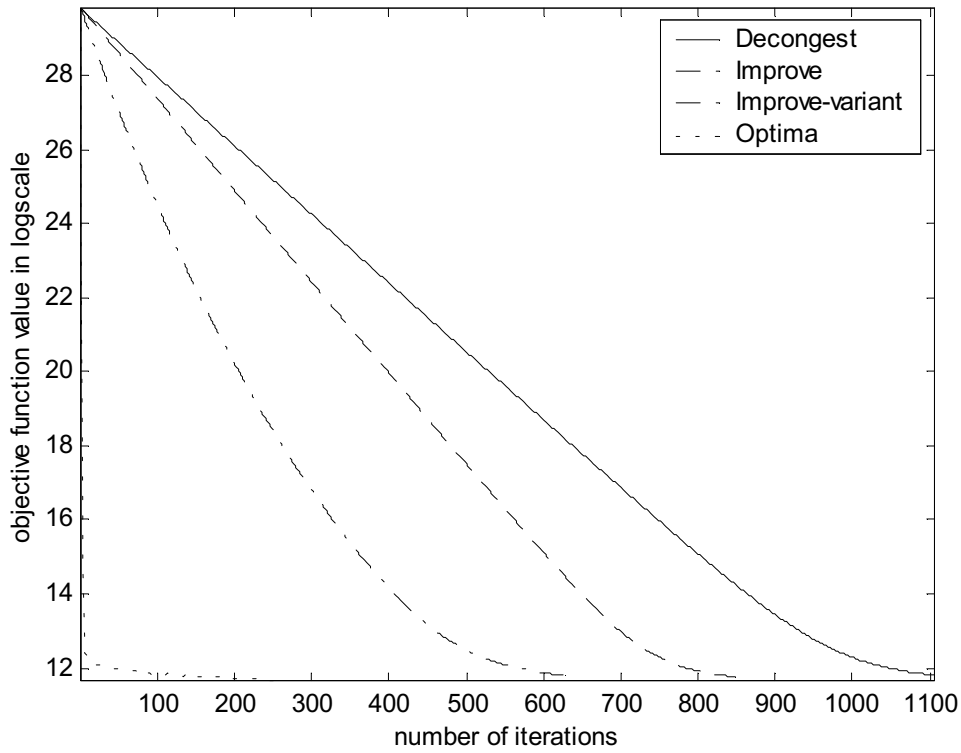

Figure 9. Objective values analysis for ε =0.1, n=100, m=1000, k=200

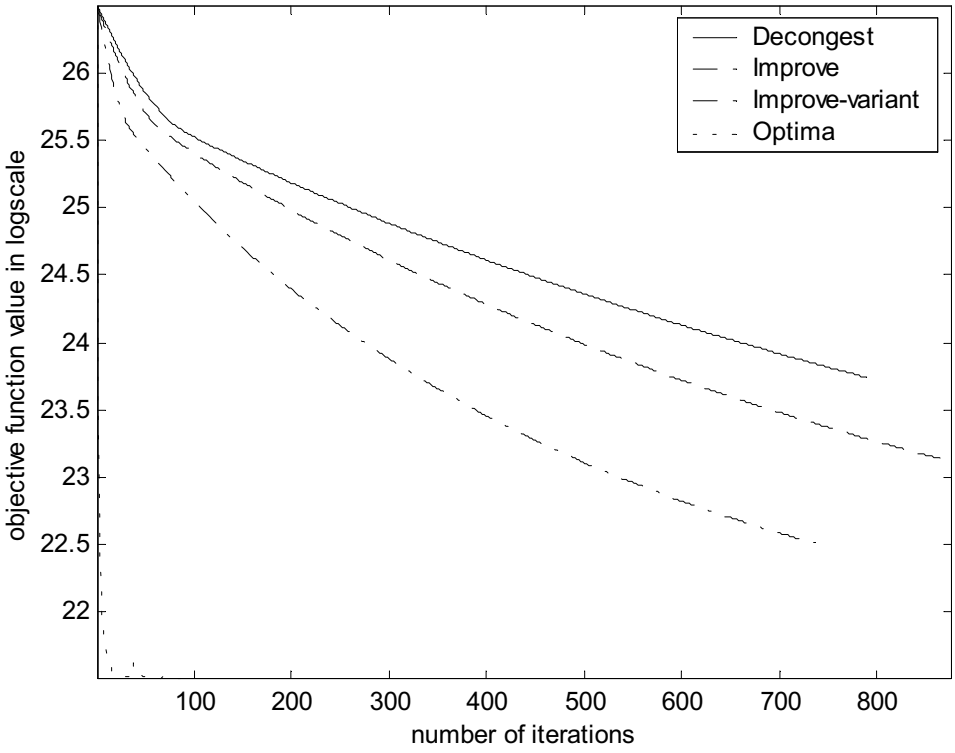

Figure 10. Objective values analysis for ε =0.1, n=200, m=600, k=500

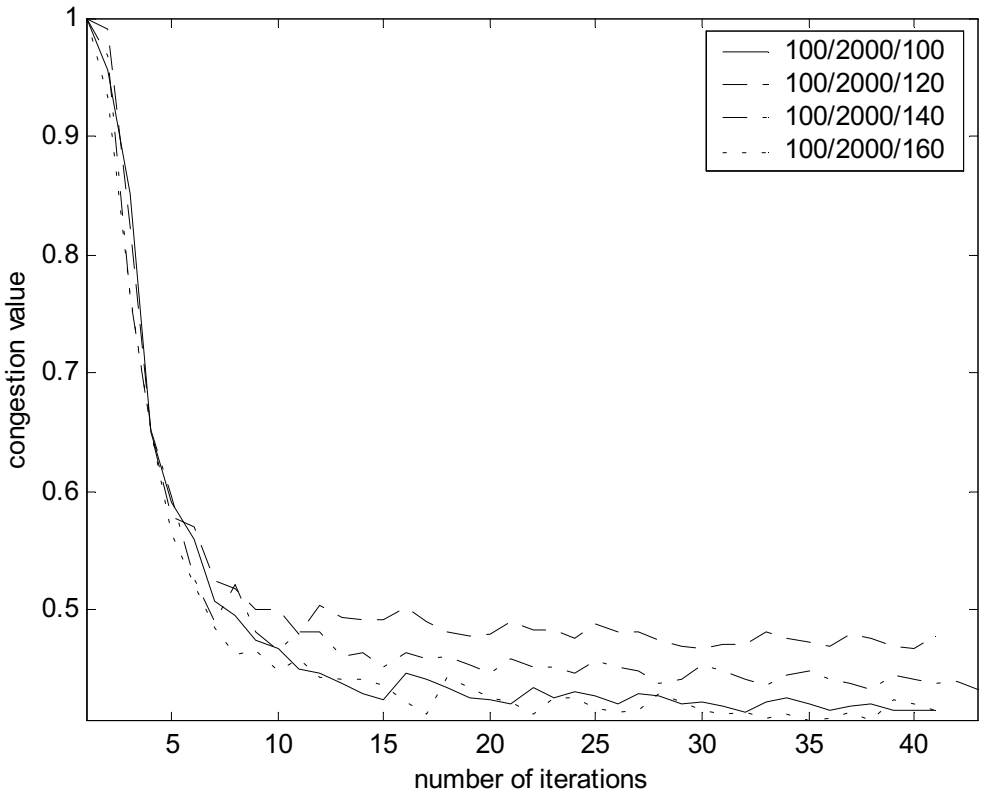

Figure 11. Congestion values analysis for transportation networks conducted by Optima

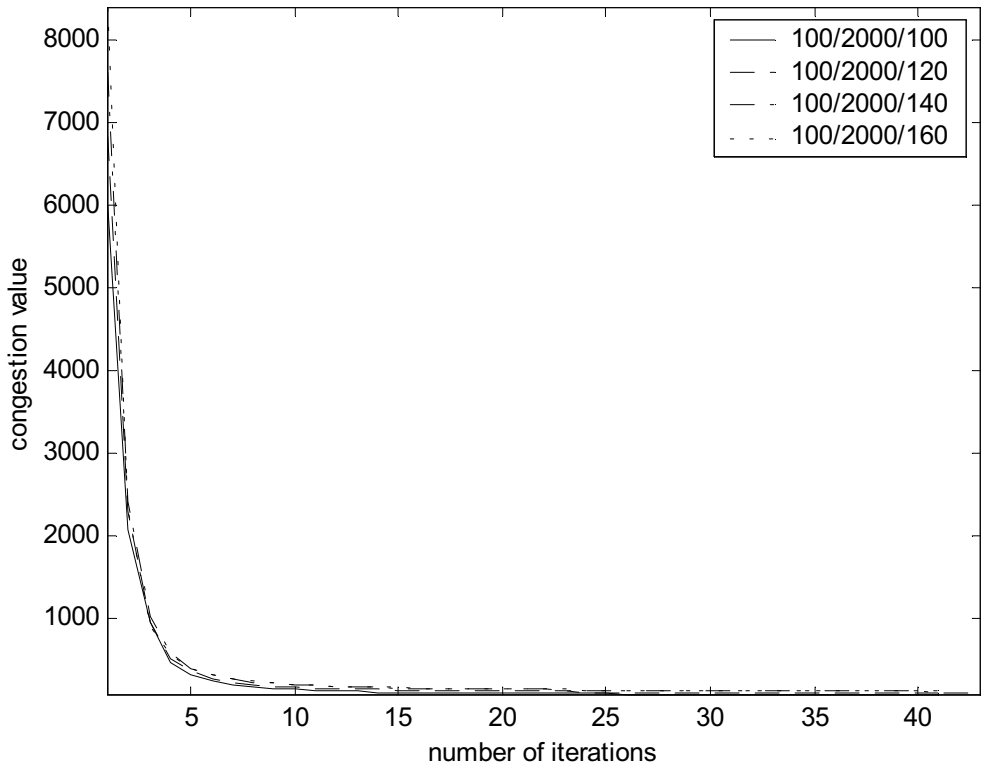

Figure 12. Congestion values analysis for transportation networks conducted by Optima