Computer and Information Science; Vol. 8, No. 2; 2015
ISSN 1913-8989 E-ISSN 1913-8997
Published by Canadian Center of Science and Education

Comparing Algorithms for Minimizing Congestion and Cost in the Multi-Commodity k-Splittable Flow

Chengwen Jiao[1], Suixiang Gao[1] & Wenguo Yang[1]

[1] School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing, China

Correspondence: Wenguo Yang, School of Mathematical Sciences, University of Chinese Academy of Sciences, Beijing, 101408, China. E-mail: yangwg@ucas.ac.cn

Received: January 19, 2015 Accepted: March 13, 2015 Online Published: April 6, 2015

doi:10.5539/cis.v8n2p1 URL: http://dx.doi.org/10.5539/cis.v8n2p1

Abstract

In the k-splittable flow problem, each commodity can only use at most k paths and the key point is to find the suitable transmitting paths for each commodity. To guarantee the efficiency of the network, minimizing congestion is important, but it is not enough, the cost consumed by the network is also needed to minimize. Most researches restrict to congestion or cost, but not the both. In this paper, we consider the bi-objective (minimize congestion, minimize cost) k-splittable problem. We propose three different heuristic algorithms for this problem, A_1, A_2 and A_3. Each algorithm finds paths for each commodity in a feasible splittable flow, and the only difference between these algorithms is the initial feasible flow. We compare the three algorithms by testing instances, showing that choosing suitable initial feasible flow is important for obtaining good results.

Keywords: k-splittable flow, minimum congestion, minimum cost, heuristic algorithm

1. Introduction

In the traditional multi-commodity flow problems, the number of paths each commodity can use is not restricted. While in practice, large number of paths may reduce the central management of the network. Specifically in the multi-protocol label switched (MPLS) networks, data packets are transmitted by the label switched paths (LSPs) that support the routing of data traffic between different terminal nodes. Large number of LSPs will decrease the performance of the protocol. Baier (2005) proposed the k-splittable flow problem and the only difference from the traditional multi-commodity flow problem is that the number of paths each commodity can use is restricted. The k-splittable flow problem can be described as follows: Given a directed graph $G=(V,E,u,c)$ with node set V and edge set E. Each edge $e\in E$ has an arc capacity $u_e>0$ and an arc cost $c_e>0$. A set of commodities is denoted by L, each commodity $l\in L$ has a certain amount of demand d_l to transmit from a source node s_l to a destination node t_l. The number of paths commodity l can use is k^l. If $k^l=1, \forall l\in L$, it is the unsplittable flow problem (UFP) which is introduced by Kleinberg (1996). If $k^l\geq|E|, \forall l\in L$, it resolves into the traditional multi-commodity flow problem.

For the k-splittable flow problem, researchers generalized the four optimization versions introduced by Kleinberg (1996) for the UFP. These versions are minimum congestion, minimum number of rounds, maximum routable demand and maximum concurrent flow. In this paper, the minimum congestion version is studied and the aim is to find the smallest value $\alpha>0$ such that when using at most α fraction of the capacity of each edge there still exists a k-splittable flow satisfying all demands.

Baier et al. (2005) proved that the maximum single commodity k-splittable flow problem is strongly NP-hard for directed graphs and they designed approximating algorithms to solve the maximum budget constrained single- and multi-commodity k-splittable flow problem. Koch et al. (2008) considered the maximum single commodity k-splittable problem as a two-stage problem, where the first stage determined the k paths and then determined the amount of the flow value on each path in the second stage. Kolliopoulos (2005) studied the minimum-cost single-source 2-splittable flow problem with the assumption that the minimum edge capacity is **larger than or equal to** the maximum commodity demand. The author proposed an approximation algorithm using rounding down strategy with factor (2, 1) for minimum congestion and cost. Salazar et al. (2006) considered the single source k-splittable flow problem. They used rounding up strategy and designed an approximation algorithm with factor (1+1/k+1/(2k-1), 1) for minimum congestion and cost under the same assumption as Kolliopoulos (2005).

Truffot et al. (2005, 2007, 2008) and Gamst et al. (2010, 2012, 2013) used branch-and-price to design algorithms to solve the single- and multi-commodity k-splittable flow problems exactly. Branch-and-price combines column generation and branching strategies to solve large scale mixed integer linear programs. But these algorithms cannot obtain exact solutions in short time which are not suitable for the high speed networks.

Caramia et al. (2008) proposed an exact algorithm based on branch-and-bound rules to solve the maximum concurrent k-splittable flow problem. The authors designed a fast heuristic algorithm for the same problem (Caramia, 2010). The commodities are first routed using an augmenting path algorithm and then a local search routine re-routes part of the paths. Jiao et al. (2014) considered the minimum congestion of the single source multi-commodity flow problem in the MPLS networks and designed fast heuristic algorithms.

Except some approximation algorithms, such as Kolliopoulos (2005) and Salazar et al. (2006), there is little study on the algorithms for the bi-objective k-splittable problem of minimizing congestion and cost. Solely minimizing the congestion may increase the total cost of the network, since commodities may use long paths to reduce the congestion. While not considering the congestion, only reducing the cost, commodities may concentrate on using the lower-cost edges which will overload the network drastically. For example, in Figure 1, a graph is given and a commodity with demand 1 from s to t is requested. Two edge-disjoint paths are used to transmit the commodity. Path P_1 has $N>1$ edges, each edge in P_1 has cost $C>1$, and path P_2 has only one edge with cost 1, the capacities of all edges equal to 1. If we only minimize the congestion, the flow value of each path is 1/2, and the total cost is $1/2NC+1/2$, the congestion is the minimum value 1/2. If we only minimize the total cost, the whole demand will be transmitted on path P_2, with minimum cost 1 but maximum congestion 1. This is not a good thing for the real network with such congestion, since other commodities cannot use these edges with congestion values already 1. We hope to find a compromise way to assess the congestion and cost.

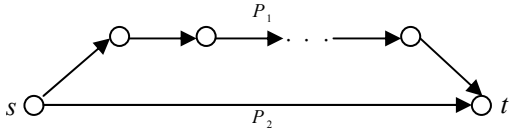

Figure 1. A commodity with demand value 1 from s to t

In this paper, we consider the bi-objective k-splittable flow problem. The demands and path restrictions of all commodities must be satisfied, and the objective function is a convex combination of normalized congestion and cost. In Section 2 we will describe the mathematical model in details. We propose heuristic algorithms in Section 3 and the simulation results are presented in Section 4.

2. Mathematical Formulation

Given a directed graph $G=(V,E,u,c)$ as before, the set of commodities is denoted by L, each commodity $l\in L$ has four parameters (s_l, t_l, d_l, k^l), meaning that commodity l can only use at most k^l paths to transmit d_l flow from s_l to t_l. The set of all feasible paths of commodity l in G is denoted by P^l. $c_p^l=\sum_{e\in p}c_e$ denotes the unit cost of the path $p\in P^l$.

Let variable x_p^l denotes the flow value on path $p\in P^l$, $y_p^l\in\{0,1\}$ denotes whether or not path $p\in P^l$ is used by commodity l, if $y_p^l=1$, path p is used by commodity l, otherwise not. $u_p:=\min\{u_e: e\in p\}$ denotes the maximum flow value that path p can transmit. The mathematical model can be formulated as follows, denote it by (MI).

$$\text{(MI)}\qquad \min\ \lambda\cdot\alpha/Cong_{opt}+(1-\lambda)\cdot\sum_{l\in L}\sum_{p\in P^l}c_p^l\cdot x_p^l/Cost_{opt} \tag{1}$$

$$s.t.\qquad \sum_{l\in L}\sum_{p\in P^l}\delta_e^p\cdot x_p^l\le\alpha\cdot u_e,\quad \forall e\in E \tag{2}$$

$$\sum_{p\in P^l}x_p^l=d_l,\quad \forall l\in L \tag{3}$$

$$x_p^l\le y_p^l\cdot M\cdot u_p,\quad \forall l\in L,\ p\in P^l \tag{4}$$

$$\sum_{p\in P^l}y_p^l\le k^l,\quad \forall l\in L \tag{5}$$

$$x_p^l\ge 0,\quad \forall l\in L,\ p\in P^l \tag{6}$$

$$y_p^l\in\{0,1\},\quad \forall l\in L,\ p\in P^l \tag{7}$$

$$\alpha>0, \tag{8}$$

The objective function (1) is to minimize the convex combination of two ratios, $\lambda \in [0,1]$ is a constant. α and $\sum_{l\in L}\sum_{p\in P^l} c_p^l x_p^l$ are the congestion and cost values of the network, respectively. We normalize the congestion and cost in the objective function by dividing their corresponding optimal values, $Cong_{opt}$ and $Cost_{opt}$, respectively. $Cong_{opt}$ and $Cost_{opt}$ are defined as follows:

When $\lambda = 1$, the objective function stresses the congestion and not considering the cost. We denote the optimal minimum congestion in this case by $Cong_{opt}$ .When $\lambda = 0$, replace constraints (2) and (4) by $\sum_{l\in L}\sum_{p\in P^l} \delta_e^p x_p^l \le u_e,\ \forall e\in E$ (2') and $x_p^l \le y_p^l \cdot u_p,\ \forall l\in L,\ p\in P^l$ (4'), respectively. Furthermore, we suppose that there is a feasible solution under the new constraints. In this case, the objective function **emphasizes** the cost, not considering the congestion . We denote the optimal minimum cost by $Cost_{opt}$.

The first constraints (2) ensure that the flow value on edge e is at most α of its capacity, $\delta_e^p \in \{0,1\}$ is a constant, if $e\in p$, $\delta_e^p = 1$, otherwise 0. The constraints (3) ensure that each commodity's demand is satisfied. Constraints (4) indicate that only path p is used by commodity l, that is $y_p^l = 1$, the flow value x_p^l can be non-negative. The constant M is any upper bound of the congestion value, which can be selected by $\sum_{l\in L} d_l \setminus u_{\min}$ with $u_{\min} := \min\{u_e : e\in E\}$. Constraints (5) limit the number of paths each commodity can use.

Constraints (6) - (8) force the variables to take on feasible values.

Although we describe the above mathematical model exactly, it is difficult to solve it directly. First, it is a mixed integer linear program which is NP-hard to solve. Second, the path set P^l denotes all the feasible paths of commodity l in G and it is not an easy thing to obtain it entirely. Third, the number of paths in P^l increases exponentially with the network's increasing, and so the size of variables is increasing drastically which will run out of the memory of computer. Finally, obtaining $Cong_{opt}$ and $Cost_{opt}$ is not an easy thing.

In practice, obtaining exact solutions is time consuming which is not allowed in the high speed transportation networks. The fast heuristic algorithms are more preferable. Choosing the suitable limited number of paths in the k-splittable flow problem is critical for the congestion and cost of the network. In the next section, we propose simple algorithms that quickly find paths for each commodity from a feasible splittable flow that satisfies all demands. We test instances in section 4 to compare the effectiveness of the algorithms.

3. Heuristic Algorithms

Before describing the heuristic algorithms, we first give three relaxed linear programs, namely (R_1), (R_2) and (R_3). Each of the three programs finds a feasible splittable flow (not considering the number of paths each commodity use) that satisfies the demands of all the commodities respect to different objective functions. The objective of (R_1) is to minimize the congestion of the network. Suppose that the optimal objective value of (R_1) is less than or equal to 1, denote it by $Cong_{lb}$, it is easy to see that $Cong_{lb} \le Cong_{opt}$. Constraints (11) ensure that each commodity's demand is satisfied and (12) are the flow conservations. $A^+(v)$ and $A^-(v)$ denote the out-going and in-going arcs of node $v\in V$, respectively. The objective of (R_2) is to minimize the total cost under the condition that all edge capacities are satisfied, see (16) , denote the optimal cost value by $Cost_{lb}$. Similarly, we have $Cost_{lb} \le Cost_{opt}$. The objective of (R_3) is the same as that of (R_2) and the congestion of each edge is not greater than $Cong_{lb}$, see (18).

$$(R_1) \qquad \min\ \alpha \tag{9}$$

$$s.t.\quad \sum_{l\in L} x_e^l \le \alpha \cdot u_e, \qquad \forall e\in E \tag{10}$$

$$\sum_{e\in A^+(s_l)} x_e^l - \sum_{e\in A^-(s_l)} x_e^l = d_l, \qquad \forall l\in L \tag{11}$$

$$\sum_{e\in A^+(v)} x_e^l - \sum_{e\in A^-(v)} x_e^l = 0, \qquad \forall l\in L,\ v\in V,\ v\ne s_l, t_l \tag{12}$$

$$x_e^l \geq 0, \qquad \forall l \in L,\ \forall e \in E \tag{13}$$

$$\alpha > 0 \tag{14}$$

$$(R_2) \qquad \min \sum_{e\in E}\sum_{l\in L} c_e \cdot x_e^l \tag{15}$$

$$s.t. \quad \sum_{l\in L} x_e^l \leq u_e, \qquad \forall e \in E \tag{16}$$

Constraints (11)-(13)

$$(R_3) \qquad \min \sum_{e\in E}\sum_{l\in L} c_e \cdot x_e^l \tag{17}$$

$$s.t. \quad \sum_{l\in L} x_e^l \leq Cong_{lb} \cdot u_e, \qquad \forall e \in E \tag{18}$$

Constraints (11)-(13)

The main idea of the heuristic algorithm we will design is as follows: Firstly, find a feasible splittable flow that satisfies all the demands of the commodities. The splittable flow can be decomposed into $|L|$ sub-flows, one for each commodity; Secondly, find limited number of transmitting paths from the flow for each commodity; Finally, allocate flow value to each path through a simple linear program with the appropriate objective function.

The sketch of the heuristic algorithm is as follows:

Step 1: Find a feasible splittable flow $f_{initial}$ that satisfies all the demands of the commodities in L. The flow $f_{initial}$ can be decomposed into $|L|$ sub-flows, that is $f_{initial} = \sum_{l\in L} f_l$ with f_l being a flow of value d_l from s_l to t_l.

Step 2: For $l = 1,...,|L|$, let $f := f_{l-1} + f_l$, define $f_0 = 0$, find paths from f for commodity l that carry the largest flow values iteratively. Once a path is found, delete its edge flow values from f (for simplicity, we also denote the current remaining flow by f), and then find the next one, until the total flow value of paths already found for commodity l is equal to d_l or the number of paths is equal to k^l. Denote the paths found for commodity l by R^l. At the end of the l-th iteration, update f_l by $f_l := f$.

Step 3: Reallocate flow values to each path in R^l for commodity l with the objective function

$$\lambda \cdot \alpha / Cong_{lb} + (1-\lambda) \cdot \sum_{l\in L}\sum_{p\in R^l} c_p^l \cdot x_p^l / Cost_{lb} \tag{19}$$

It is in fact to solve the following linear program, denote it by (R_4). $\delta_e^p \in \{0,1\}$ is a constant, if $e \in p$, $\delta_e^p = 1$, otherwise $\delta_e^p = 0$.

$$(R_4) \qquad \min\ \lambda \cdot \alpha / Cong_{lb} + (1-\lambda) \cdot \sum_{l\in L}\sum_{p\in R^l} c_p^l \cdot x_p^l / Cost_{lb} \tag{20}$$

$$s.t. \quad \sum_{l\in L}\sum_{p\in R^l} \delta_e^p \cdot x_p^l \leq \alpha \cdot u_e, \quad \forall e \in E \tag{21}$$

$$\sum_{p\in R^l} x_p^l = d_l, \qquad \forall l \in L \tag{22}$$

$$x_p^l \geq 0, \qquad \forall l \in L,\ p \in R^l \tag{23}$$

$$\alpha > 0 \tag{24}$$

For simplicity, in the objective function of (R_4) we use the lower bounds, $Cong_{lb}$ and $Cost_{lb}$ of the optimal congestion and cost values $Cong_{opt}$, $Cost_{opt}$. The above linear problem is easy to solve, since $|R^l| \leq k^l$, the number of variables is largely reduced compared to (MI).

In step 2, at the beginning of dealing with the next commodity, say l, we not only use its initial flow f_l but also the remaining flow for the previous $l-1$ iterations, say f_{l-1}. We find paths in $f_{l-1} + f_l$ for commodity l. By this way, we can get better paths for the current commodity.

We can see that given different initial feasible flows, we can obtain different algorithms. From solving the three linear programs, (R_1), (R_2) and (R_3), we get three different initial feasible splittable flows, and hence three heuristic algorithms are obtained, denote them by A_1, A_2 and A_3, respectively. Since the initial flow in A_3 is obtained by (R_3) with network's congestion less than or equal to $Cong_{lb}$, that is from all the feasible splittable flows with congestion value less than or equal to $Cong_{lb}$, we choose the one with minimum cost, and the flow is tend to use the edges with small cost values. We guess that A_3 may have a better compromise effectiveness in congestion and cost, while A_1 has good effectiveness in low congestion and A_2 has advantage in low cost. To show this, we test 72 instances in the next section.

4. Computational Results

We use the Carbin instances, Bl instances and Bs instances, as our testing instances. All the instances have 32 nodes, the number of edges is between 96 and 320, and the number of commodities is between 48 and 320. All tests have been performed with uniform values of k, i.e. $k^l = k$ for all commodities in L. We use three different values of k, that are 1, 2 and 5. For simplicity, we only test instances on the case $\lambda = 1$ of Step 3 in the heuristic algorithms. After solving (R_4), we compute the resulting total cost of the transmitting paths. Tests were performed on an Intel Core 2.4 GHz processor, 4 GB of RAM. We use CPLEX to solve the linear programs. The experimental results are reported on Table1-Table4 in the Appendix. As for the time spent on the tests of the three algorithms, it has little differences and the most time-consuming part is obtaining the initial feasible splittable flows in Step 1. In this paper, we omit the time results and mainly analyze the results of the congestion and cost values of the three algorithms.

In Table1 and Table2, for each instance name, the first column followed is the number of paths each commodity can use, the next three columns are the congestions obtained by A_1, A_2 and A_3, respectively, and the last three columns are the ratios between the congestion values and their optimal lower bounds, $Cong_{lb}$. For the test results of the 36 Bl instances in Table1, we can see that all the 36 congestions obtained by A_1 is less than or equal to that of A_2, 35 of 36 congestions obtained by A_1 is less than or equal to that of A_3, and 34 of 36 congestions obtained by A_3 is less than or equal to that of A_2. We conclude that A_1 has advantage in congestion, and A_2 is the poorest among the three algorithms. As for the test results of the 36 Bs instances in Table2, we have similar results.

In Table3 and Table4, we list the test results for the three algorithms on cost. For each instance name, the first column followed is the number of paths each commodity can use, the next three columns are the ratios between the cost values and their optimal lower bounds, $Cost_{lb}$. From Table3, we can see that all the cost values obtained by A_2 is less than or equal to that of A_1, 35 of 36 cost values obtained by A_2 is less than or equal to that of A_3, and all the cost values obtained by A_3 is less than or equal to that of A_1. We conclude that A_2 has advantage in cost, and A_1 is the poorest among the three algorithms. As for the test results for the 36 Bs instances in Table4, we have similar results.

From the above analysis, we can see that algorithm A_1 performed best in congestion but worst in cost and algorithm A_2 performed best in cost but worst in congestion. It is intelligible since A_1 uses the feasible splittable flow that is obtained from (R_1) which restricts on congestion but not cost. Similarly, A_2 uses the feasible splittable flow that is obtained from (R_2) which restricts on cost but not congestion. As for the algorithm A_3, the congestion and cost values are between that of A_1 and A_2. The congestion values of A_3 are closer to that of A_1 than A_2, and the cost values of A_3 are closer to that of A_2 than A_1. A_3 has a good compromise between congestion and cost. We conclude that not considering cost in the beginning will cause a high cost, such as A_1, while not considering congestion will cause a high congestion, such as A_2.

5. Conclusions

In this paper, we consider the bi-objective (minimize congestion, minimize cost) k-splittable flow problem. We propose the mathematical model for this problem and use a convex combination of the normalized congestion and cost as the objective function. We propose the sketch of a kind of heuristic algorithms which begins with a feasible splittable flow satisfying all the demands of the commodities. This kind of algorithms is suitable for the general multi-source k-splittable flow problem. We compare three different heuristic algorithms through testing instances and they show different advantages on the effectiveness of congestion and cost. As for this kind of heuristic algorithms, the most time-consuming part is to get the initial feasible splittable flow which is obtained by solving a linear program. If the size of commodity set is large enough, we need to design other fast algorithms which rely on the feasible flows less. In the future, we will continue to study the bi-objective k-splittable problem and design better algorithms.

Acknowledgements

This work is supported by the National 973 Plan project under Grant No. 2011CB706900，the National 863 Plan project under Grant No.2011AA01A102, the NSF of China (11331012, 71171189), the "Strategic Priority Research Program" of the Chinese Academy of Sciences(XDA06010302), and Huawei Technology Co. Ltd.

References

Baier, G., Köhler, E., & Skutella, M. (2005). On the k-splittable flow problem. *Algorithmica, 42*, 231-248. http://dx.doi.org/10.1007/s00453-005-1167-9

Caramia, M., & Sgalambro, A. (2008). An exact approach for the maximum concurrent k-splittable flow problem. *Optimization Letters, 2*, 251-265. http://dx.doi.org/10.1007/s11590-007-0055-4

Caramia, M., & Sgalambro, A. (2010). A fast heuristic algorithm for the maximum concurrent k-splittable flow problem. *Optimization Letters, 4*, 37-55. http://dx.doi.org/10.1007/s11590-009-0147-4

Gamst, M. (2013). A decomposition based on path sets for the multi-commodity k-splittable Maximum Flow Problem. Department of Management Engineering, Technical University of Denmark, DTU Management Engineering Report No.6.

Gamst, M., & Petersen, B. (2012). Comparing branch-and-price algorithms for the multicommodity k-splittable maximum flow problem. *European Journal of Operational Research, 217*(2), 278-286. http://dx.doi.org/10.1016/j.ejor.2011.10.001

Gamst, M., Jensen, P. N., Pisinger, D., & Plum, C. (2010). Two-and three-index formulations of the mini-mum cost multicommodity k-splittable flow problem. *European Journal of Operational Research, 202*(1), 82-89. http://dx.doi.org/10.1016/j.ejor.2009.05.014

Jiao, Ch. W., Yang, W. G., Gao, S. X., Xia, Y. B., & Zhu, M. M. (2014). The k-Splittable Flow Model and a Heuristic Algorithm for Minimizing Congestion in the MPLS Networks. Inter-national Conference on Natural Computation(ICNC), Xiamen University, 19-21 August 2014.

Jiao, Ch.W., Gao, S. X., Yang, W. G., Xia, Y. B., & Zhu, M. M. (2014). A Fast Heuristic Algorithm for Minimizing Congestion in the MPLS Networks. *Int.J. Communications, Network and System Sciences, 7,* 294-302. http://dx.doi.org/10.4236/ijcns.2014.78032

Kleinberg, J. M. (1996). Single-source unsplittable flow. *In Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, 68-77.

Koch, R., Skutella, M., & Spenke, I. (2008). Maximum k-splittable s,t-flows, *Theory of Computing Systems*, *43*(1), 1432-4350. http://dx.doi.org/10.1007/s00224-007-9068-8

Kolliopoulos, S. G. (2005). Minimum-cost single-source 2-splittable flow. *Information Processing Letters, 94*(1), 15-18. http://dx.doi.org/10.1016/j.ipl.2004.12.009

Salazar, F., & Skutella, M. (2006). Single-source k-splittable min-cost flows. *Operations research letters, 37,* 71-74. http://dx.doi.org/10.1016/j.orl.2008.12.004

Truffot, J., & Duhamel, C. (2008). A branch and price algorithm for the k-splittable maximum flow problem. *Discrete Optimization, 5*(3), 629-646. http://dx.doi.org/10.1016/j.disopt.2008.01.002

Truffot, J., Duhamel, C., & Mahey, P. (2005). Using branch-and-price to solve multicommodity k-splittable flow problem. *The Proceedings of International Network Optimization Conference (INOC)*, Lisbonne, 20-23, March 2005.

Truffot, J., Duhamel, C., & Mahey, P. (2007). K-Splittable Delay Constrained Routing Problem: A Branch and Price Approach. *Design and Reliable Communication Networks (DRCN), 6th International Workshop on,* La Rochelle, 7-10. http://dx.doi.org/10.1109/DRCN.2007.4762284

Appendix

Table 1. Congestion results for the three algorithms on the Carbin instances called *Bl*

instance	k	A_1	A_2	A_3	$Gaps_1$	$Gaps_2$	$Gaps_3$
Bl01	1	1.308	1.342	1.302	1.323	1.358	1.318
Bl01	2	0.988	1.109	1.023	1.000	1.031	1.035
Bl01	5	0.988	0.999	0.992	1.000	1.011	1.004
Bl03	1	1.464	1.882	1.619	1.485	1.909	1.642
Bl03	2	0.986	1.010	1.010	1.000	1.024	1.024
Bl03	5	0.986	0.995	0.991	1.000	1.009	1.005
Bl05	1	0.340	1.619	0.941	1.395	6.647	3.864
Bl05	2	0.253	1.088	0.389	1.040	4.465	1.597
Bl05	5	0.244	0.952	0.246	1.000	3.910	1.012
Bl07	1	0.512	1.450	1.133	1.554	4.402	3.441
Bl07	2	0.413	1.000	0.531	1.254	3.036	1.613
Bl07	5	0.329	1.000	0.339	1.000	3.036	1.030
Bl09	1	1.133	1.385	1.529	1.155	1.412	1.559
Bl09	2	1.000	1.012	1.013	1.019	1.032	1.033

Bl09	5	1.000	1.003	1.000	1.019	1.023	1.019
Bl11	1	1.131	1.197	1.197	1.145	1.212	1.212
Bl11	2	0.993	0.994	0.993	1.005	1.006	1.005
Bl11	5	0.993	0.994	0.993	1.005	1.006	1.005
Bl13	1	1.250	1.818	1.539	1.651	2.401	2.032
Bl13	2	0.763	1.107	1.000	1.007	1.462	1.321
Bl13	5	0.757	1.000	0.813	1.000	1.321	1.073
Bl15	1	1.000	1.524	1.286	1.495	2.279	1.923
Bl15	2	0.714	1.083	0.794	1.068	1.620	1.188
Bl15	5	0.695	1.083	0.733	1.039	1.620	1.097
Bl17	1	1.152	1.220	1.220	1.158	1.226	1.226
Bl17	2	0.995	1.014	1.014	1.000	1.019	1.019
Bl17	5	0.995	1.014	1.014	1.000	1.019	1.019
Bl19	1	1.250	1.250	1.250	1.256	1.256	1.256
Bl19	2	0.995	0.999	0.995	1.000	1.004	1.000
Bl19	5	0.995	0.999	0.995	1.000	1.004	1.000
Bl21	1	1.231	1.474	1.462	1.858	2.225	2.206
Bl21	2	0.686	1.043	0.870	1.036	1.574	1.313
Bl21	5	0.677	1.000	0.677	1.021	1.510	1.021
Bl23	1	1.250	1.667	1.539	1.715	2.287	2.111
Bl23	2	0.787	1.031	0.800	1.080	1.415	1.098
Bl23	5	0.729	1.031	0.786	1.000	1.415	1.078

Table 2. Congestion results for the three algorithms on the Carbin instances called *Bs*

instance	k	A_1	A_2	A_3	$Gaps_1$	$Gaps_2$	$Gaps_3$
Bs01	1	1.800	1.588	1.385	1.841	1.624	1.416
Bs01	2	0.990	1.048	0.987	1.013	1.009	1.072
Bs01	5	0.978	1.000	0.980	1.000	1.023	1.003
Bs03	1	1.857	1.345	1.345	1.885	1.365	1.365
Bs03	2	0.986	1.059	1.059	1.000	1.074	1.074
Bs03	5	0.986	1.000	1.000	1.000	1.015	1.015
Bs05	1	1.357	2.333	1.357	2.797	4.808	2.797
Bs05	2	0.679	1.111	0.744	1.398	2.290	1.534
Bs05	5	0.485	1.000	0.500	1.000	2.061	1.030
Bs07	1	1.188	1.682	1.267	2.824	4.000	3.013
Bs07	2	0.679	1.154	0.679	1.614	2.744	1.614
Bs07	5	0.421	1.000	0.448	1.000	2.378	1.066
Bs09	1	1.333	1.333	1.333	1.333	1.333	1.333
Bs09	2	1.000	1.017	1.017	1.000	1.017	1.017
Bs09	5	1.000	1.000	1.000	1.000	1.000	1.000
Bs11	1	1.170	1.281	1.300	1.187	1.300	1.319
Bs11	2	0.990	0.996	0.990	1.004	1.011	1.005
Bs11	5	0.990	0.998	0.990	1.004	1.013	1.004
Bs13	1	1.462	1.750	1.429	2.448	2.931	2.393
Bs13	2	0.694	1.200	0.740	1.162	2.010	1.240
Bs13	5	0.625	1.024	0.625	1.047	1.715	1.047
Bs15	1	1.429	1.750	1.667	2.271	2.782	2.650
Bs15	2	0.776	1.011	0.791	1.233	1.607	1.257
Bs15	5	0.654	1.000	0.654	1.039	1.590	1.039
Bs17	1	1.059	1.786	1.786	1.104	1.861	1.861
Bs17	2	0.974	1.003	0.974	1.015	1.045	1.015
Bs17	5	0.959	1.000	0.959	1.000	1.042	1.000
Bs19	1	1.215	1.240	1.240	1.224	1.248	1.248
Bs19	2	0.993	1.003	1.000	1.000	1.009	1.007
Bs19	5	0.993	1.003	1.000	1.000	1.009	1.007

Bs21	1	1.429	1.933	1.429	2.110	2.855	2.110
Bs21	2	0.775	1.012	0.776	1.144	1.494	1.146
Bs21	5	0.692	1.002	0.704	1.022	1.480	1.039
Bs23	1	1.188	1.667	1.308	1.969	2.764	2.168
Bs23	2	0.702	1.027	0.657	1.164	1.703	1.090
Bs23	5	0.614	1.004	0.632	1.047	1.665	1.047

Table 3. Cost results for the three algorithms on the Carbin instance called *Bl*

instance	k	$C-Gaps_1$	$C-Gaps_2$	$C-Gaps_3$	instance	k	$C-Gaps_1$	$C-Gaps_2$	$C-Gaps_3$
Bl01	1	1.179	0.996	0.994	Bl13	1	1.583	0.998	1.104
Bl01	2	1.155	1.008	1.009	Bl13	2	1.532	1.020	1.098
Bl01	5	1.145	1.005	1.009	Bl13	5	1.507	1.025	1.102
Bl03	1	1.130	1.003	1.016	Bl15	1	1.476	1.013	1.133
Bl03	2	1.119	1.011	1.022	Bl15	2	1.406	1.019	1.136
Bl03	5	1.098	1.021	1.032	Bl15	5	1.400	1.026	1.147
Bl05	1	2.298	0.997	1.309	Bl17	1	1.085	1.004	1.012
Bl05	2	2.375	1.007	1.364	Bl17	2	1.052	1.004	1.006
Bl05	5	2.539	1.022	1.340	Bl17	5	1.060	1.004	1.006
Bl07	1	2.548	1.007	1.192	Bl19	1	1.073	1.008	1.016
Bl07	2	2.272	1.014	1.132	Bl19	2	1.073	1.010	1.013
Bl07	5	2.377	1.014	1.162	Bl19	5	1.069	1.008	1.012
Bl09	1	1.108	1.000	1.015	Bl21	1	1.317	1.000	1.147
Bl09	2	1.060	1.006	1.011	Bl21	2	1.249	1.023	1.145
Bl09	5	1.061	1.010	1.018	Bl21	5	1.233	1.027	1.145
Bl11	1	1.080	0.999	1.001	Bl23	1	1.361	1.010	1.103
Bl11	2	1.047	1.011	1.011	Bl23	2	1.278	1.015	1.102
Bl11	5	1.038	1.009	1.010	Bl23	5	1.278	1.016	1.100

Table 4. Cost results for the three algorithms on the Carbin instance called *Bs*

instance	k	$C-Gaps_1$	$C-Gaps_2$	$C-Gaps_3$	instance	k	$C-Gaps_1$	$C-Gaps_2$	$C-Gaps_3$
Bs01	1	1.044	1.044	1.017	Bs13	1	1.424	1.010	1.420
Bs01	2	1.050	1.013	1.023	Bs13	2	1.332	1.018	1.294
Bs01	5	1.052	1.017	1.023	Bs13	5	1.308	1.029	1.288
Bs03	1	1.017	0.993	0.995	Bs15	1	1.412	1.010	1.201
Bs03	2	1.060	1.010	1.010	Bs15	2	1.279	1.027	1.175
Bs03	5	1.062	1.010	1.011	Bs15	5	1.265	1.024	1.176
Bs05	1	2.219	1.010	1.405	Bs17	1	1.158	1.013	1.040
Bs05	2	2.606	1.050	1.474	Bs17	2	1.101	1.007	1.033
Bs05	5	2.454	1.046	1.550	Bs17	5	1.093	1.006	1.030
Bs07	1	1.941	0.993	1.526	Bs19	1	1.096	1.011	1.006
Bs07	2	1.854	1.017	1.486	Bs19	2	1.093	1.011	1.010
Bs07	5	1.862	1.036	1.516	Bs19	5	1.089	1.011	1.011
Bs09	1	1.097	1.014	1.014	Bs21	1	1.379	1.011	1.206
Bs09	2	1.060	1.003	1.003	Bs21	2	1.304	1.016	1.208
Bs09	5	1.059	1.006	1.006	Bs21	5	1.299	1.016	1.206
Bs11	1	1.084	1.004	1.019	Bs23	1	1.375	1.002	1.345
Bs11	2	1.063	1.012	1.014	Bs23	2	1.301	1.017	1.294
Bs11	5	1.069	1.005	1.023	Bs23	5	1.279	1.019	1.279