Shipper Collaboration

Özlem Ergun
Gültekin Kuyzu
Martin Savelsbergh

Georgia Institute of Technology
Industrial and Systems Engineering
Atlanta, GA 30332

Abstract

The interest in collaborative logistics is fuelled by the ever increasing pressure on companies to operate more efficiently, the realization that suppliers, consumers, and even competitors, can be potential collaborative partners in logistics, and the connectivity provided by the Internet. Logistics exchanges or collaborative logistics networks use the Internet as a common computing platform to implement strategies designed to reduce "hidden costs" such as asset reposition costs. Through collaboration shippers may be able to identify and submit tours with little or no asset repositioning to a carrier, as opposed to submitting individual lanes, in return for more favorable rates. In this paper, we focus on finding a set of tours connecting regularly executed truckload shipments so as to minimize asset repositioning. Mathematically, the truckload shipper collaboration problem translates into covering a subset of arcs in a directed Euclidean graph by a minimum cost set of constrained cycles. We formulate the lane covering problem, propose several solution algorithms, and conduct a computational study on the effectiveness of these methodologies.

1 Introduction

The growing interest in collaborative logistics is fuelled by the ever increasing pressure on companies to operate more efficiently; the realization that suppliers, consumers, and even competitors, can be potential collaborative partners in logistics; and the connectivity provided by the Internet.

In the trucking industry, both shippers and carriers are desperately searching for ways to operate more efficiently. Shippers are under continuous pressure from the market place to increase logistics performance while reducing costs. They have to move smaller product quantities more often to meet shorter customer lead times. Furthermore, the buffers that once protected against shortages have shrunk, as inventory volumes grow smaller. Mistakes that were once covered by excess inventory now emerge as expedited logistics costs. Carriers are facing challenges to profitability too. Margins are at an all-time low and driver turnover is at an all-time high. To make matters worse, fuel costs are surging and carrier insurance costs are on the rise as well.

Traditionally shippers and carriers have focused their attention on controlling their own costs to increase profitability, i.e., improve those business processes that the organization controls independently. The key to collaborative logistics lies in identifying and reducing "hidden costs" that all participants in a logistics system pay, but none control individually. An example of a hidden cost is asset repositioning. To execute shipments from different shippers a carrier often has to reposition its assets. Shippers have no insight in how the interaction between the various shipments affects a carrier's asset repositioning costs. In other words, no single participant in the logistics system controls the asset repositioning costs. Asset repositioning is expensive. A recent report estimates that 18% of all trucks movements every day are empty. In a $921 billion U.S. logistics market, the collective loss is staggering: more than $165 billion.

Logistic exchanges or collaborative logistics networks, such as those offered and managed by Nistevo, Elogex, and Transplace use the Internet as a common computing platform to give shippers and carriers visibility to hidden costs. Through collaboration, participants of logistics networks can implement strategies specifically designed to reduce or eliminate these hidden costs.

A good example that reveals the benefits of collaboration involves scheduling truckload movements of multiple shippers. Through collaboration, two members of the Nistevo network created and are using a dedicated 2,500-mile continuous move route, reaching several Midwestern and Eastern U.S. cities, involving seven stages, and encompassing various distribution centers, production facilities, and retail outlets for both companies. The route has little asset repositioning and gives the carrier's drivers a repeatable schedule. The route resulted in a combined 19% savings for both shippers. The carrier is experiencing higher margins and lower driver turnover through more regular driver schedules.

When shippers consider collaborating, their goal is to identify sets of lanes that can be

submitted to a carrier as a bundle, rather than individually, in the hope that this results in more favorable rates. Carriers are often willing to provide more favorable rates for bundles when a bundle of lanes provides repeatable work for a driver and covering the lanes in the bundle involves little or no asset repositioning. The shipper collaboration problem can thus be stated as follows: given a set of lanes, find a set of tours that covers all lanes and that minimizes the asset repositioning.

We will focus on the simplest variant, which is static and involves only full truckloads. This setting is relevant for companies that regularly send full truckloads, say once a week, and are looking for collaborative partners in similar situations. The underlying assumption is that shipment schedules can be adjusted so that the resulting tours can be executed in practice.

Identifying tours that minimize asset repositioning costs in a collaborative logistics network is not easy. As the number of participants in the network, hence the number of truckload movements, grows the number of potential routes to examine becomes prohibitively large. Optimization technology is needed to assist the logistics network provider's analysts in identifying tours with little or no asset repositioning.

In practice, there will be various restrictions limiting the set of acceptable tours such as a restriction on the maximum number of legs that can make up a tour or on the maximum length of a tour. Our initial analysis shows that such constraints make the underlying optimization problem more difficult, from a theoretical perspective, but they may make it easier to construct high quality solutions, from a practical perspective. In this paper, we study the core optimization model arising in this context with restrictions on the number of legs that can make up a tour and develop and computationally test effective and efficient algorithms to solve this optimization problem.

2 Problem Definition and Complexity

2.1 Lane Covering Problem

Mathematically, the shipper collaboration problem translates to the *Lane Covering Problem* (LCP). Given a directed Euclidean graph $D = (N, A)$ with node set N, arc set A, arc costs $c(a)$ for $a \in A$, and lane set $L \subseteq A$, find a minimum cost set of simple cycles covering L (a cycle cover of L). Note that an arc may be traversed more than once and thus may appear in multiple cycles.

Theorem 1 *LCP can be solved in polynomial time.*

Proof: We will present two algorithms that solve the LCP in polynomial time:

1. Consider the digraph D as described above. Assign an upper bound of infinity to all the arcs in A, a lower bound of 1 to all the arcs in L and 0 for all the arcs in

$A \backslash L$. Now let C be a minimum cost circulation on this graph. The subgraph D' of D induced by all the arcs with positive flow in C is a Eulerian graph. Furthermore since all the arcs in L have a positive flow in C the subgraph D' includes all the lane arcs. Hence the flow C on D' can be decomposed into unit flow on directed cycles covering all arcs in L providing a minimum cost lane cover.

2. Now we will describe another algorithm which resembles the algorithm for the Chinese Postman Problem [3].

   Let $D(L)$ be the subgraph defined by the arcs in L and let $U = \{v \in D(L) : deg^+(v) > deg^-(v)\}$ and $V = \{v \in D(L) : deg^+(v) < deg^-(v)\}$. Define a complete bipartite network $\mathcal{N}$ with partitions U and V. For each arc $a = uv$ of $\mathcal{N}$ let the capacity be $|L|$ and the weight be the length of the shortest di-path between u and v in D. Assign to each node $v \in U$ a demand of $deg^+(v) - deg^-(v)$ and to each node $v \in V$ a supply of $deg^-(v) - deg^+(v)$. Solve the min-cost flow problem in $\mathcal{N}$.

   Let F denote the set of arcs in $\mathcal{N}$ to which a flow is assigned in the optimal solution. For each arc $a = uv$ let $D(a)$ be the set of arcs of D corresponding to the shortest di-path from u to v in D. Let $T = \bigcup_{a \in F} D(a)$. In case of repetitions, keep multiple copies.

   Observe that $D' = (N, L \cup T)$ satisfies $deg^+(v) = deg^-(v), \forall v \in N$. Hence, every component of D' is a directed Euler tour. Partition the Euler tours into simple arc-disjoint directed cycles. These cycles define a minimum cost cycle cover of L. □

2.2 Constrained Lane Covering Problems

In this section we consider two constrained variants of the lane covering problem: (i) *the cardinality constrained lane covering problem* (CCLCP), where the cardinality of each cycle, i.e., the number of arcs in the cycle, is less than or equal to a prespecified number K, and (ii) *the length constrained lane covering problem* (LCLCP), where the length of the cycle is less than or equal to a prespecified bound B.

Unfortunately, both LCLCP and CCLCP are more difficult. We will show the NP-hardness result for the LCLCP by a reduction from the three partition problem. The result for the CCLCP follows from a similar but more detailed reduction.

Theorem 2 *LCLCP is NP-hard.*

Proof: The proof is by reduction from 3-Partitioning.

3-Partitioning. *Given a set* $S = \{a_1, a_2, ..., a_{3n}\}$ *with* $\frac{1}{4}B < a_i < \frac{1}{2}B$ *for* $i = 1, ..., 3n$ *and* $B = \sum_{i=1}^{3n} a_i/n$ *decide whether there exist a partitioning of* S *into distinct subsets* S_k *for* $k = 1, ..., n$ *such that* $\sum_{i \in S_k} a_i = B$.

Construct a directed graph $D = (V, A)$ as follows. For each a_i create three arcs e_1^i, e_2^i, e_3^i with length a_i. The arcs are placed in three layers, L_1, L_2, and L_3. Every arc e_k^i is connected to arc $e^j_{(k \bmod 3)+1}$ for all $j \neq i$ for $k = 1, 2, 3$ by an arc of length $K >> B$.

Observe that:

- Any cycle with length less than $B + 3K$ will have at most 6 arcs.
- Any cycle with at most 6 arcs can only contain a single arc from $\{e_1^i, e_2^i, e_3^i\}$.
- A set $\{e_1^i, e_2^i, e_3^i\}$ of arcs can be covered by three "identical" cycles.

¿From these observations it follows easily that there exists a cycle cover of cost $3n(B + 3K)$ with cycles of length less than $B + 3K$ if and only if the instance of 3-partitioning has a feasible solution. □

Theorem 3 *CCLCP is NP-hard.*

Proof: This follows immediately from the previous proof by replacing each arc e_i^k of length a_i by a_i copies of length 1 and noticing that 3-Partition is strongly NP-complete. □

3 Literature

We have been unable to find any literature on LCP, but there does exist a body of research on a related covering problem. The cycle covering problem (CCP) looks for a least cost cover of a graph with simple cycles, each containing at least three different edges. This constrained version of the Chinese Postman Problem (CPP) was shown to be NP-hard on general graphs by Thomassen [10] and to be equivalent to the CPP on planar graphs by Kesel'man [8]. Itai et al. [6] provided an upper bound for CCP on 2-connected unweighted graphs and gave a polynomial time algorithm which finds such a cover. Improvements to this bound were proposed by Bermond et al. [2], Alon and Tarsi [1], Fraisse [5], Jackson [7], and Fan [4], and a simple heuristic was proposed and tested by Labbe et al. [9].

4 Solution Approaches for the CCLCP

We will focus on the cardinality constrained lane covering problem in the rest of the paper.

4.1 Set Covering Formulation

First we formulate the CCLCP as a set covering problem. Let $\mathcal{C}_K$ represent the set of all directed cycles in D of cardinality less than or equal to K. We may assume that each cycle $C \in \mathcal{C}_K$ is such that $C \cap L \neq \emptyset$. Let c_C denote the cost of cycle C, l_C be 1 if lane l is on the cycle C, and x_C be a 0-1 variable indicating whether cycle C is selected or not. Then CCLCP can be formulated as the following set covering problem

$$\min \sum_{C \in \mathcal{C}_K} c_C x_C \tag{1}$$

$$\sum_{C \in \mathcal{C}_K} l_C x_C \geq 1 \quad \forall l \in L \tag{2}$$

$$x_C \in \{0,1\} \quad \forall C \in \mathcal{C}_K. \tag{3}$$

4.2 Greedy Algorithm

A simple greedy algorithm can be develop based on this set covering formulation of CCLCP, where in each iteration a cycle C is chosen that maximizes the "cover factor" of a cycle, i.e., the ratio of the length of the lanes covered by the cycle and the total length of the cycle.

---

procedure Greedy
begin
$U := L$;
Until $U = \emptyset$ **do**:
Choose $C \in \mathcal{C}_K$ such that $C \cap U \neq \emptyset$ and such that $\frac{\sum_{e \in C \cap U} c(e)}{c_C}$ is maximal;
$U := U \setminus (C \cap L)$;
end

Theorem 4 *Assuming that if $(i,j) \in L$ then $(j,i) \in A$ and $c(i,j) = c(j,i)$, the greedy algorithm is a 2-approximation algorithm.*

Proof: The algorithm produces a solution within a factor 2 of the optimum. Each lane $(i,j) \in L$ can be covered with a trivial cycle consisting of the arcs (i,j) and (j,i). The cover factor of such a cycle is 0.5. The greedy algorithm always has the option of choosing such a trivial 2-cycle to cover a lane. Hence the greedy will never construct a solution with value more than 2 times $\sum_{e \in L} c(e)$ which is a lower bound on the optimal solution. □

In fact, we conjecture that the greedy algorithm is a 1.5-approximation algorithm.

Conjecture 1 *Assuming that if* $(i,j) \in L$ *then* $(j,i) \in A$ *and* $c(i,j) = c(j,i)$*, the greedy algorithm is a 1.5-approximation algorithm.*

There exists an instance for which this bound is tight. Consider a complete graph $D = (N, A)$ with $N = \{1, 2, 3, 1', 2', 3'\}$, and let $K = 3$. The lane set, L, consists of six lanes $L = \{(1,2), (2,3), (3,1), (1,3'), (3,2'), (2,1')\}$ (Figure 1(a)), where the length of each lane arc is equal to 1 and $c(i,i') = \epsilon$ for $i = 1, 2, 3$. The optimal solution with cost $6 + 3\epsilon$ consists of three cycles $\{(1,2,1'), (3,1,3'), (2,3,2')\}$ (Figure 1(c)). However the greedy algorithm will pick the cycle $(1,2,3)$ in the first iteration and hence has to complete the lane cover with three additional trivial cycles $\{(2,1'), (1,3'), (3,2')\}$ (Figure 1(b)). This solution has total cost equal to 9 which becomes 1.5 times the optimal value as ϵ goes to 0.

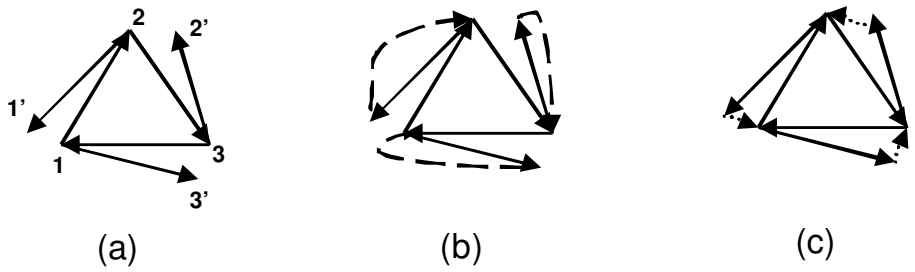

Figure 1: Illustrating a 1.5-approximate solution generated by the greedy algorithm

Note that for any fixed K the algorithm runs in polynomial time since $|\mathcal{C}_K| = o(|L|^K)$ and at most $|L|$ iterations are required to reach the stopping condition.

5 Computational Experiments

A key component of both solution approaches discussed above is the generation of cycles. There is an important trade-off to be made. The more cycles we generate, the better is the quality of the solution we obtain. However, even though polynomially bounded, the number of cycles becomes prohibitively large very quickly. We analyze this trade-off by generating all possible cycles, which can be very time consuming, and by generating cycles with at most one single repositioning arc, which can be done very efficiently.

LCP instances are created randomly in two steps. First, a complete Euclidean digraph is created based on three parameters: number of nodes, number of clusters, and cluster

factor. Clusters are introduced to represent metropolitan areas or other geographical clusters of points. Let *short arcs* be arcs connecting points within a cluster and *long arcs* be arcs connecting points in different clusters. The cluster factor determines the radius of the cluster (with a smaller factor leading to a shorter radius). Second, the lane set is created based on two parameters: number of lanes and short percentage p. We randomly select $p\%$ arcs from the set of short arcs and the rest from the set of long arcs.

The results presented below are based on three different digraphs, each with 500 nodes. The digraphs differ in terms of the number of clusters and cluster factor. We chose: 20 clusters and cluster factor 0.05 for digraph 1, 10 clusters and cluster factor 0.1 for digraph 2, and 5 clusters and cluster factor 0.1 for digraph 3. Furthermore, in all experiments we used a short percentage of 20, i.e., 20% of the lanes are short and 80% of the lanes are long, and we restricted the cardinality of a cycle to be at most 5. Therefore, the maximum number of repositioning arcs in a cycle is at most 2 since we assume that all of our digraphs are complete. All computational experiments were run on a 3Ghz Pentium 4 processor with 1Gb of memory.

Our first set of computational results, presented in Table 1, shows the value of working with the set of all possible cycles as opposed to working with the set of cycles containing at most a single repositioning arc. We report the number of repositioning arcs allowed (#dh), the sum of the length of the cycles selected by the Greedy algorithm (greedy), the ratio of reposition length and cycle length (ratio), and the percentage improvement in terms of total cycle length when there is no restriction on the number of repositioning arcs in the cycle. Note that a small ratio is preferred over a large ratio.

There are two important conclusions that can be drawn from these results. First, as the number of lanes increases the quality of the solution increases as well (the repositioning ratio decreases). This is not surprising because the number of opportunities for cycles with little repositioning should increase with the number of lanes (note that the underlying digraph remains the same). Second, as expected, there is a significant improvement in quality (about 25-30%) when we consider all cycles rather than just the cycles with a single deadhead arc. Observe too that the improvement is the largest for instances with the fewest number of lanes.

Our second set of computational results, presented in Table 2, demonstrates the quality of the solutions produced by the greedy heuristic by comparing it to the solution produced by solving the set partitioning problem. We have used XPRESS-MP 2003 for the solution of the integer programs. We have stopped at the first feasible integer solution because for the larger instances it took too long to solve the problem to optimality. However, we observed that in most cases the first feasible integer solution is actually optimal. For the largest instances, i.e., no restriction on the number of repositioning arcs in a cycle and 1500 lanes, the integer program was too large to fit into the memory available.

There are two important conclusions that can be drawn from these results. First, the quality of the solution produced by the greedy heuristic is excellent, usually within

Table 1: Generating cycles with a single deadhead vs. generating all cycles

digraph	#lanes	#dh	greedy	ratio	#dh	greedy	ratio	%
1	100	1	10,674,528.92	0.47	2	7,241,893.35	0.22	32
1	200	1	19,797,922.49	0.44	2	13,797,469.95	0.19	30
1	500	1	44,763,006.66	0.36	2	31,963,459.64	0.10	29
1	1000	1	83,391,224.24	0.32	2	60,134,487.56	0.06	28
1	1500	1	115,360,691.20	0.28	2	88,152,466.22	0.06	24
2	100	1	10,930,492.56	0.45	2	6,872,427.22	0.13	37
2	200	1	20,435,489.90	0.42	2	13,163,089.15	0.11	36
2	500	1	45,443,680.97	0.37	2	32,170,219.78	0.11	29
2	1000	1	80,549,668.78	0.31	2	61,359,625.85	0.09	24
2	1500	1	117,287,520.87	0.28	2	91,198,131.37	0.07	22
3	100	1	9,669,176.32	0.48	2	5,605,234.45	0.10	42
3	200	1	18,159,907.33	0.42	2	12,031,403.25	0.12	34
3	500	1	40,116,888.68	0.38	2	27,623,007.27	0.10	31
3	1000	1	74,816,084.92	0.33	2	52,065,742.43	0.04	30
3	1500	1	108,007,282.59	0.29	2	80,297,771.09	0.04	26

a few percent of optimality. Second, when the number of cycles to choose from is very large, i.e., when there is no restriction on the number of repositioning arcs in a cycle, the performance of greedy heuristic is exceptionally good (within 12% of optimality).

Our final set of computational results, presented in Table 3, shows the computational requirements of cycle generation, the greedy heuristic, and the integer program. There are two key observations that can be made. First, the number of feasible cycles increases rapidly with the number of lanes, especially if we do not limit the number of repositioning arcs in the cycles; with 1500 lanes the number of cycles with cardinality at most 5 already exceeds 4,000,000. In such situations, cycle generation time become prohibitively large. Second, the computational requirements of the greedy heuristic are minimal; even when the number of cycles exceeds 4,000,000 the computation time is only about 15 seconds. Finally, the time required to solve integer program increases rapidly as the number of cycles increases.

6 Conclusions

The large-scale adoption of the internet as a medium for information sharing is creating new opportunities for improved logistics performance. One of the promising opportuni-

Table 2: Quality of Greedy heuristic vs. Set Partitioning

digraph	#lanes	#dh	greedy	ip	%
1	100	1	10,674,528.92	10,657,287.67	0.16
1	200	1	19,797,922.49	19,796,100.57	0.01
1	500	1	44,763,006.66	43,933,356.60	1.85
1	1000	1	83,391,224.24	80,442,234.07	3.54
1	1500	1	115,360,691.20	109,195,127.20	5.34
1	100	2	7,241,893.35	7,152,454.23	1.24
1	200	2	13,797,469.95	13,485,151.01	2.26
1	500	2	31,963,459.64	31,440,110.74	1.64
1	1000	2	60,134,487.56	59,533,223.89	1.00
1	1500	2	88,152,466.22	-	-
2	100	1	10,930,492.56	10,930,492.56	0.00
2	200	1	20,435,489.90	20,415,042.13	0.10
2	500	1	45,443,680.97	44,786,656.84	1.45
2	1000	1	80,549,668.78	77,691,955.13	3.55
2	1500	1	117,287,520.87	110,141,137.60	6.09
2	100	2	6,872,427.22	6,733,012.72	2.03
2	200	2	13,163,089.15	12,977,248.41	1.41
2	500	2	32,170,219.78	31,793,064.81	1.17
2	1000	2	61,359,625.85	60,975,233.15	0.63
2	1500	2	91,198,131.37	-	-
3	100	1	9,669,176.32	9,610,905.39	0.60
3	200	1	18,159,907.33	18,114,536.24	0.25
3	500	1	40,116,888.68	39,843,407.61	0.68
3	1000	1	74,816,084.92	70,995,026.42	5.11
3	1500	1	108,007,282.59	100,383,352.00	7.06
3	100	2	5,605,234.45	5,510,851.62	1.68
3	200	2	12,031,403.25	11,968,537.69	0.52
3	500	2	27,623,007.27	27,539,192.17	0.30
3	1000	2	52,065,742.43	51,963,016.61	0.20
3	1500	2	80,297,771.09	-	-

Table 3: Number of cycles and solution times

digraph	#lanes	#dh	#cycles	cycle cpu	greedy cpu	ip cpu
1	100	1	119	0.00	0.00	0.00
1	200	1	270	0.00	0.00	0.00
1	500	1	1,123	0.00	0.00	0.00
1	1000	1	4,966	0.01	0.00	0.00
1	1500	1	15,977	0.05	0.01	0.00
1	100	2	4,124	0.01	0.00	0.00
1	200	2	19,712	0.06	0.02	0.00
1	500	2	219,468	3.80	0.44	115.00
1	1000	2	1,380,971	155.59	4.22	5,719.00
1	1500	2	4,426,539	1654.99	15.72	-
2	100	1	114	0.00	0.00	0.00
2	200	1	275	0.00	0.00	0.00
2	500	1	1,179	0.00	0.00	0.00
2	1000	1	5,507	0.01	0.00	0.00
2	1500	1	17,502	0.05	0.02	0.00
2	100	2	4,032	0.01	0.00	0.00
2	200	2	21,082	0.06	0.03	0.00
2	500	2	220,506	3.48	0.44	132.00
2	1000	2	1,493,193	177.75	4.44	5,784.00
2	1500	2	4,653,239	1812.84	16.98	-
3	100	1	122	0.00	0.00	0.00
3	200	1	275	0.00	0.00	0.00
3	500	1	1,021	0.00	0.00	0.00
3	1000	1	4,310	0.01	0.00	0.00
3	1500	1	14,041	0.03	0.03	0.00
3	100	2	4,491	0.01	0.00	0.00
3	200	2	20,914	0.06	0.02	0.00
3	500	2	194,061	2.73	0.36	104.00
3	1000	2	1,283,701	131.48	3.63	8,244.00
3	1500	2	4,246,370	1494.28	14.75	-

ties is collaborative logistics. We have started to examine one problem in this realm: shipper collaboration. To develop effective technology to support shipper collaboration a challenging, but interesting, combinatorial optimization problem needs to be solved: the lane covering problem. We have demonstrated that efficient heuristic solution procedures can be implemented that provide high quality solutions. The key challenge in terms of future developments is to either reduce the time required to generate the cycles, or to develop methodology that does not require all cycles to be generated. We are currently pursuing the latter approach, by developing local search procedures that start from a solution consisting of cycles with at most a single reposition arc (which we can generate very efficiently) and improving it by allowing the creation of cycles with more than a single repositioning arc.

Acknowledgement

We thank Daniel Espinoza and Marcos Goycoolea for stimulating and insightful discussions.

References

[**1**] N. Alon and M. Tarsi, Covering multigraphs by simple circuits, SIAM Journal on Algorithmic and Discrete Methods 6 (1985) 345-350.

[**2**] J.C. Bermond, B. Jackson, and F. Jaeger, Shortest covering of graphs with cycles, Journal of Combinatorial Theory, Series B 35 (1983) 297-308.

[**3**] H.A. Eiselt, M. Gendreau and G. Laporte, Arc routing problems, part 1: the Chinese postman problem, Operations Research 43 (1995) 231-242.

[**4**] G. Fan, Covering graphs by cycles, SIAM Journal on Discrete Mathematics 5 (1992) 491-496.

[**5**] P. Fraisse, Cycle covering in bridgeless graphs, Journal of Combinatorial Theory, Series B 39 (1985) 146-152.

[**6**] A. Itai, R.J. Lipton, C.H. Papadimitriou, and M. Rodeh, Covering graphs by simple circuits, SIAM Journal on Computing 10 (1981) 746-750.

[**7**] B. Jackson, Shortest circuit covers and postman tours in graphs with a nowhere zero 4-flows, SIAM Journal on Computing 19 (1990) 659-665.

[**8**] D.Y. Kesel'man, Covering the edges of a graph by circuits, Kibernetica 3 (1987) 16-22.

[**9**] M. Labbe, G. Laporte, and P. Soriano, Covering a graph with cycles, Computers and Operations Research 25 (1998) 499-504.

[**10**] C. Thomassen, On the complexity of finding a minimum cycle cover of a graph, SIAM Journal on Computing 26 (1997) 675-677.