A Two-Stage Solution Approach for the Directed Rural Postman Problem with Turn Penalties

Carmine Cerrone*
Department of Biosciences and Territory
University of Molise
carmine.cerrone@unimol.it

Benjamin Dussault
End-to-End Analytics, LLC
Palo Alto, CA 94303
ben.dussault@gmail.com

Xingyin Wang
Engineering Systems and Design
Singapore University of
Technology and Design
xingyin_wang@sutd.edu.sg

Bruce Golden
Robert H. Smith School of Business
University of Maryland
bgolden@rhsmith.umd.edu

Edward Wasil
Kogod School of Business
American University
ewasil@american.edu

June 30, 2018

Abstract

In this paper, we consider the Directed Rural Postman Problem with Turn Penalties (DRPP-TP). A solution is a tour that traverses all required arcs of the graph. The total cost of the tour is the sum of the lengths of the traversed arcs plus the penalties associated with the turns. One solution approach involves transforming the arc routing problem into an equivalent node routing problem. An alternative direct approach (without graph transformation) that involves two stages has been proposed in the literature. In the first part of this paper, we investigate the applicability of the direct approach. We identify several characteristics of the input instance that make this approach effective and present several limitations of this approach. In the second part of this paper, we describe an integer linear program that is combined with a local search algorithm. This combination produces high-quality solutions to the DRPP-TP in a reasonable amount of computing time.

Keywords. Routing, Heuristics, Greedy Algorithm, Rural Postman Problem, Turn Penalties.

* Corresponding author

1 Introduction

The Directed Rural Postman Problem (DRPP) [8, 10] is an important arc routing problem with many real-world applications including street sweeping, meter reading, postal delivery, and snow plowing. In some of these applications, the quality of a tour is determined not only by the length of the tour but also by the types of turns that are made at street intersections. For example, most truck drivers prefer to travel straight ahead for as long as possible [16]. Turning left or even turning right can be dangerous and time consuming. U-turns are impossible for long trucks. In snow plowing, left turns and U-turns are often discouraged because they take more time and push snow into an intersection. In these applications, we have to take into account the cost of turning when solving the problem.

In this paper, we consider the Directed Rural Postman Problem with Turn Penalties (DRPP-TP). In the DRPP-TP, we need to find the tour that traverses all arcs in a set of required arcs in a directed graph while minimizing the length of the tour and the sum of penalties associated with the turns. This problem was introduced by Benavent and Soler [2]. The authors claim that, in contrast to the DRPP, the DRPP-TP remains NP-Hard even if all arcs are required. Of course, when all arcs are required, the DRPP reduces to a Directed Chinese Postman Problem (DCPP) which is solvable in polynomial time. The authors developed a heuristic for the DRPP-TP based on the results of Bodin and Kursh [4]. Finally, they proved that the DRPP-TP can be solved by transforming it to an asymmetric traveling salesman problem (TSP) defined on a graph where the number of nodes is equal to the number of required arcs.

Since 1999, several papers [1, 5, 11, 12, 17, 21] have been published that focus on routing problems with turn penalties. In these papers, well-known routing problems are generalized to take turn penalties into account. Theoretical results about resolution and complexity were provided for these problems. Exact and heuristic approaches produced high-quality computational results.

Transforming the input graph into a new graph with the costs of the turns in its structure is common to solution approaches. Soler et al. [20] proposed a transformation into an asymmetric TSP. They provided exact solutions to several instances. Irnich [14] proposed a method based on a transformation to the asymmetric TSP. This transformation modeled turn and street crossing restrictions, cluster constraints, and alternative service modes, such as zigzag service. Micó and Soler [18] transformed an input graph into a generalized vehicle routing problem and then solved the problem both exactly and heuristically. Clossey et al. [9] developed an alternative direct approach for turn penalties that used two stages. In the first stage, the problem was solved as a rural postman problem to obtain an Eulerian graph. In the second stage, an end-pairing algorithm was used to generate an Eulerian tour taking into consideration the turn penalties. The authors defined six different strategies for the end-pairing algorithm and found two strategies (denoted by $v2$ and $v4$) to be effective. In this paper, we refer to the direct approach (DA) as a class of algorithms that are based on the Eulerian tour-first turn-second idea. In the first half of this paper, we investigate the applicability of DA to solve the DRPP-TP. In the second half, we improve the DA of Clossey et al. [9].

The remainder of this paper is structured as follows. In Section 2, we identify a set of characteristics that makes DA effective. We compare DA to an exact approach and identify a set of features for the input graph that makes DA effective. Although DA has been described in the

literature, our work helps to determine when the approach is applicable.

In Sections 3 and 4, we focus our attention on improving DA. In particular, we provide a way to identify an Eulerian tour taking into consideration the turn penalties for an Eulerian graph. We develop an integer linear program (denoted by β-ILP) that produces an Eulerian graph taking into account turn penalties from a directed graph. We show that DA with β-ILP in the first stage and our heuristic approach in the second stage produce high-quality solutions for all of our test instances. We compare the computational results of our algorithm with the results of strategy $v4$ proposed by Clossey et al. [9], CONS2+IMP proposed by Benavent and Soler [2], and BK proposed by Bodin et al. [3]. We also perform experiments on very large real street network graphs. In all the experiments, our algorithm produces effective solutions within reasonable running times. In Section 5, we give our conclusions.

2 Applicability of the direct approach

In this section, we investigate the applicability of DA. In particular, we formulate the integer linear program model that we use in our computational experiments. We compare the results produced by the standard version of DA (i.e., both the rural postman and the end-pairing problems are solved optimally) to the optimal solutions. We identify a set of characteristics that allows widespread applicability of DA.

2.1 Flow formulation for the DRPP

We model the DRPP using a flow formulation that does not take into account the cost of turns. Let the directed graph $G = (N, A)$, where N is the set of nodes and A is the set of arcs. Let $A_R \in A$ be the subset of arcs that must be visited. The cost associated with arc $a \in A$ is denoted by $c(a)$. We define the following sets, variables and constants.

$x_a \in \mathbb{N}_{\geq 0}$ is the number of times the solution traverses arc $a \in A$.

$f_a \in \mathbb{R}_{\geq 0}$ is the flow crossing arc $a \in A$.

$N_R \in N$ is a set of nodes that are start or end points of some required arcs, i.e.,$\{i \in N \mid \exists j \in N,\ (i,j)\ or\ (j,i) \in A_R\}$.

$\delta^-(v) = \{(i,v) \in A\}$ is the set of incoming arcs into v.

$\delta^+(v) = \{(v,j) \in A\}$ is the set of outgoing arcs from v.

$M = |N_R|$.

$n_d \in N_R$ is an arbitrarily chosen node.

Our integer linear program (denoted by ILP) is given below:

(ILP) $$\text{Minimize} \sum_{a \in A} c(a) x_a \tag{1}$$

subject to

$$x_a \geq 1 \qquad \forall\, a \in A_R \tag{2}$$

$$\sum_{a \in \delta^-(i)} x_a - \sum_{a \in \delta^+(i)} x_a = 0 \qquad \forall\, i \in N \tag{3}$$

$$\sum_{a \in \delta^+(n_d)} f_a = M - 1 \tag{4}$$

$$\sum_{a \in \delta^-(i)} f_a - \sum_{a \in \delta^+(i)} f_a = 0 \qquad \forall\, i \in N \setminus N_R \tag{5}$$

$$\sum_{a \in \delta^-(i)} f_a - \sum_{a \in \delta^+(i)} f_a = 1 \qquad \forall\, i \in N_R \setminus \{n_d\} \tag{6}$$

$$f_a \leq M x_a \qquad \forall\, a \in A \tag{7}$$

$$x_a \in \mathbb{N}_{\geq 0} \qquad \forall\, a \in A \tag{8}$$

$$f_a \in \mathbb{R}_{\geq 0} \qquad \forall\, a \in A. \tag{9}$$

Constraints (2) force all required arcs in A_R to be in each solution. For each node, constraints (3) require the in-degree to be equal to the out-degree. If the solution is characterized by a single connected component, then constraints (3) are sufficient to force the variables x_a to represent an Eulerian graph. In this model, in order to guarantee the connectivity of the solution, we use a flow formulation. The node $n_d \in N_R$ is the source with supply $M-1$, where M is equal to the size of N_R. All other nodes in N_R are sinks, with a demand of one. All nodes that are not end points of any required arcs are transshipment nodes. Constraints (4), (5), and (6) force the variables f_a to represent a spanning tree rooted at n_d that connects to all nodes in N_R. Constraints (7) allow the variables f_a to be greater than zero only if the corresponding variables x_a are greater than zero. This implies that the variables x_a define a connected structure, so that constraints (3) are sufficient to define an Eulerian graph. To create the Eulerian graph, denoted by $G_e = (N_e, A_e)$, we use the values associated with the variables x_a which is the solution to the ILP. For each arc $a = (i, j) \in A$ such that $x_a > 0$, we add nodes i, j to N_e and add x_a copies of the arc (i, j) in A_e.

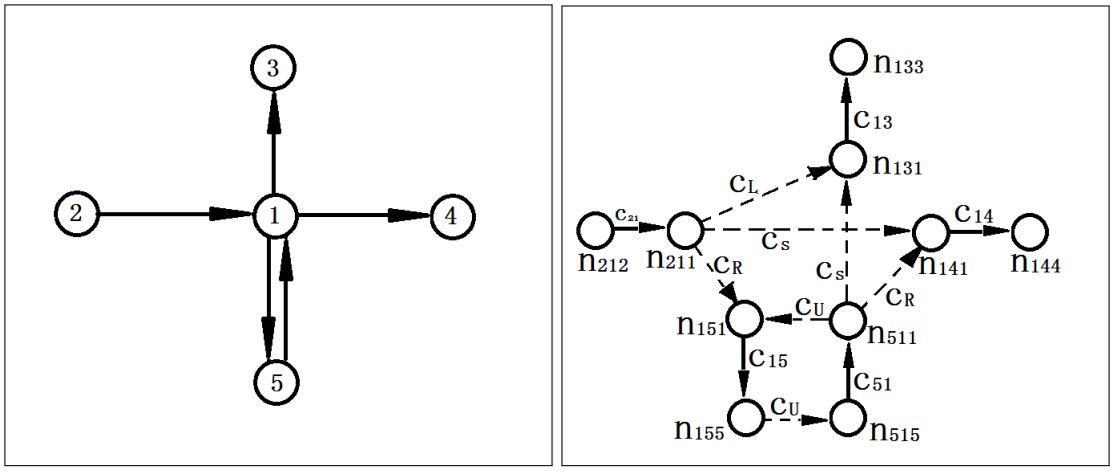
2.2 Graph transformation for the DRPP-TP

We use the graph transformation defined in [9] to solve the DRPP-TP. We produce the new graph $G' = (N', A')$ on which to apply the mathematical model. The set of arcs $A' = A'_A \cup A'_T$ is the union of arcs associated with the arcs of the original graph and the arcs associated with the turns. Specifically, for each arc $(i, j) \in A$, we insert nodes $n_{i,j,i}$ and $n_{i,j,j}$ in N' and arc $(n_{i,j,i}, n_{i,j,j})$ in A'_A. For each node $j \in N$ and pairs of arcs (i, j) and $(j, k) \in A$, we insert the arc $(n_{i,j,j}, n_{j,k,j})$ in A'_T. The cost $c(n_{i,j,i}, n_{i,j,j})$ equals $c(i, j)$, and the cost $c(n_{i,j,j}, n_{j,k,j})$ equals the cost of the turn $(i, j)(j, k)$ in G.

In addition to A'_A and A'_T, we define the following sets.

$$A'_R = \{(n_{i,j,i}, n_{i,j,j}) \in A'_A \mid (i, j) \in A_R\}.$$

$$N'_R = \{n_{i,j,k} \in N' \mid (i, j) \in A_R\}.$$



(a) (b)

Figure 1: Graph transformation example from [9].

In Fig. 1(a), we show a directed graph with five nodes and five arcs from [9]. Each arc is associated with two nodes and one arc in the transformed graph, which is shown in Fig. 1(b). For example, arc $(1, 3)$ in Fig. 1(a) is associated with nodes n_{131} and n_{133} and arc (n_{131}, n_{133}). The cost of the arc $(n_{131},\ n_{133})$, denoted by c_{13}, is the same as the cost of the arc $(1, 3)$ in the original graph. The dashed arcs in the transformed graph are associated with the turns. For example, arc (n_{211}, n_{131}) represents turn $(2, 1)$, $(1, 3)$ in the original graph. The cost of this arc, denoted by c_L, is the cost of the left turn made at node 1.

2.3 ILP for the DRPP-TP

This model (denoted by α-ILP) requires the graph G' and uses the constraints (2)-(9) with the objective function shown below:

$$(\alpha\text{-ILP}) \qquad \text{Minimize } \alpha \sum_{a \in A'_A} c(a)x_a + (1-\alpha) \sum_{a \in A'_T} \hat{c}(a)x_a. \tag{10}$$

The sets N, N_R, A, and A_R of G, used in constraints (2)-(9), are replaced by N', N'_R, A', and A'_R of G', respectively. The α-ILP, with $\alpha = 1$, 0, and 0.5, is used in the computational experiments in Sections 2.4 and 4.4.

With $\alpha = 1$, the ILP minimizes the length of the tour. In this case, the objective function is given by the minimization of $\sum_{a \in A'_A} c(a)x_a$, which is equivalent to (1).

With $\alpha = 0$, the ILP minimizes the total cost of the turns. The objective function is given by the minimization of $\sum_{a \in A'_T} \hat{c}(a)x_a$. The arcs in A'_T are related to the turns and $\hat{c}(a)$ defined $\forall\ a \in A'_T$ represents the cost of the turn a.

With $\alpha = 0.5$, the ILP produces an optimal solution for the DRPP-TP. The objective function is given by the minimization of $\frac{1}{2}(\sum_{a \in A'_A} c(a)x_a) + \frac{1}{2}(\sum_{a \in A'_T} \hat{c}(a)x_a)$. This model produces the minimum cost solution that takes into account the balance between the length of the tour and the total cost of the turns.

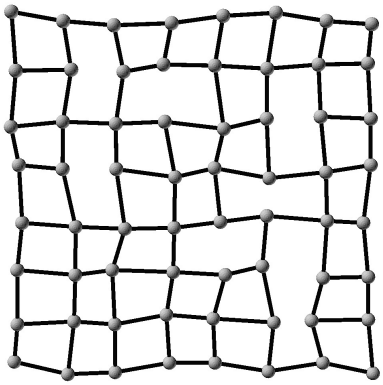


Figure 2: Example of 8×8 test instance.

2.4 Computational results

2.4.1 Test instances

Our test instances are randomly generated. Each graph represents a strongly connected grid. Each street is represented by one or two arcs depending on the direction of the street. The cost assigned to each arc is proportional to the Euclidean distance between the endpoints of the arc, where the average cost for an arc is equal to 100. Each vertex is located within 10% of its exact position on the grid in order to produce arcs with different lengths (see Fig. 2). We randomly remove a percentage of arcs from each instance and only use a graph that is totally connected. Instances are characterized by the number of nodes n (intersections) and the percentage of removed arcs p. We have three scenarios for the number of nodes ($8 \times 8, 10 \times 10, 12 \times 12$) and four scenarios for the percentage of removed arcs ($5\%, 10\%, 20\%, 30\%$). We define three scenarios for the percentage of required arcs that we need to traverse in each feasible solution ($25\%, 50\%, 75\%$ of the total number of arcs). The results reported in our tables give the average value computed on 10 graphs of the same scenario. Our codes were developed in Java 1.8 and CPLEX 12.7. All experiments were performed on an OSX 10.9.5 laptop equipped with an Intel(R) Core(TM) i7-3720QM 2.6GHz CPU and 16Gb of RAM.

In the remainder of this paper, we use the following notation. Let X be a solution to the DRPP-TP. $V(X)$ is the objective function value associated with solution X. $C(X)$, $T(X)$, $Tt(X)$, and $Ft(X)$ are the total cost, cost of the turns, number of turns, and number of prohibited turns in the solution, respectively. OPT is the optimal solution to the DRPP-TP. $DA[procA, procB]$ is the solution produced by a direct approach that uses algorithm $procA$ in the first stage and algorithm $procB$ in the second stage.

2.4.2 Results from DA compared to an optimal solution

In Tables 1 to 4, we compare the optimal solution to the results produced by DA. To produce the optimal solution to DRPP-TP, we solve α-ILP with $\alpha = 0.5$. To produce the DA solution, we run ILP on the directed graph to create the Eulerian graph G_e described in Section 2.1, and run α-ILP with $\alpha = 0$ on its transformed graph G' with all arcs required as input. To define the cost associated with each turn, we use a parameter $\gamma \in \{25, 50\}$. In the top half of Table 1 (first nine rows), we summarize the results using a turn configuration equal to $(0, \gamma, 2\gamma, 3\gamma)$.

	Multiplier γ	**25**			**50**			Average
	Required Arcs (%)	**25**	**50**	**75**	**25**	**50**	**75**	
	Removed Arcs (%)							
Turn Costs	**5**	4.84	2.29	2.45	9.87	5.91	4.54	4.98
$(0,\gamma,2\gamma,3\gamma)$	**10**	4.15	1.63	1.74	8.76	4.59	3.33	4.03
	20	2.96	1.45	1.02	5.03	3.50	1.96	2.65
	30	1.82	1.27	0.69	3.56	1.97	1.70	1.83
	Average	3.44	1.66	1.47	6.81	3.99	2.88	3.38
	Overall Avarage	2.19			4.56			
Turn Costs	**5**	1.50	0.86	0.96	3.66	1.85	2.26	1.85
$(0,\gamma,\gamma,\gamma)$	**10**	1.50	0.84	0.71	3.22	1.74	1.55	1.59
	20	0.93	0.58	0.37	2.03	1.69	0.90	1.08
	30	0.48	0.45	0.45	1.86	0.77	0.46	0.74
	Average	1.10	0.68	0.62	2.69	1.51	1.29	1.32
	Overall Avarage	0.80			1.83			

Table 1: Percentage difference in total cost between the DA solution and the optimal solution.

Using this configuration, we have costs of $(0, \gamma, 2\gamma, 3\gamma)$ for a straight turn, right turn, left turn, and U-turn, respectively. We assume a straight turn has no cost, a right turn has a unitary cost, a left turn is twice as expensive as a right turn, and a U-turn is the most costly. This scheme is a simplification of the scheme $(0, 2, 5, 9)$ proposed by Clossey et al. in [9]. Left turns are considered more dangerous than right turns in right-hand driving countries. In bottom half of Table 1, we have costs of $(0, \gamma, \gamma, \gamma)$ for a straight turn, right turn, left turn, and U-turn, respectively. In this configuration, we want to maximize the number of straight turns. The size of the input graph is fixed at 12×12.

In Table 1, we see that the gap between the exact approach and DA decreases when the density of the graph decreases, the percentage of required arcs increases, and the cost of the turns decreases. To compute the values in the table, we use the formula $100 \times \frac{V(DA[ILP,\alpha\text{-ILP}]) - V(OPT)}{V(OPT)}$.

In Tables 2, 3, and 4, we report detailed results. We use two different sets of costs for the turns $((0, \gamma, 2\gamma, 3\gamma), (0, \gamma, \gamma, \gamma))$, and three different sizes for the graphs $(8 \times 8, 10 \times 10, 12 \times 12)$. Table 2 presents the percentage difference of the optimal solution produced by α-ILP with $\alpha = 0.5$ to the solution produced by the standard DA where ILP is solved in the first stage and α-ILP with $\alpha = 0$ is solved in the second stage. In subtables **b** and **d**, we have, for a straight turn, right turn, left turn, and U-turn, the cost $(0, \gamma, \gamma, \gamma)$ (with γ equal to 50 in **b** and 25 in **d**, respectively). In these two scenarios, we maximize the number of straight turns. Each row gives the percentage of arcs that we remove from a complete grid representing our input graph. Each column gives the percentage of arcs that we need to visit in the tour. In the subtables **a** and **c**, we have, for a straight turn, right turn, left turn, and U-turn, the cost $(0, \gamma, 2\gamma, 3\gamma)$ (with γ equal to 50 in **a** and 25 in **c**, respectively). Tables 3 and 4 deal with the tour length and turn costs, respectively. DA produced better solutions in terms of tour length with respect to the optimal solution (Table 3); however, this is not true in terms of the objective function value when turn costs are taken into account (Table 2). To compute the values reported in Tables 2, 3, and 4, we use the formulas $100 \times \frac{V(DA[ILP,\alpha\text{-ILP}]) - V(OPT)}{V(OPT)}$, $100 \times \frac{C(DA[ILP,\alpha\text{-ILP}]) - C(OPT)}{C(OPT)}$, and $100 \times \frac{T(DA[ILP,\alpha\text{-ILP}]) - T(OPT)}{T(OPT)}$,

Number of Nodes			8×8			10×10			12×12		
	Required Arcs (%)		**25**	**50**	**75**	**25**	**50**	**75**	**25**	**50**	**75**
	Removed Arcs (%)										
a	$(0, \gamma, 2\gamma, 3\gamma)$ **50**	**5**	6.86	6.35	3.57	8.22	5.05	5.22	9.87	5.91	4.54
		10	7.25	4.45	2.60	7.33	4.49	3.16	8.76	4.59	3.33
		20	5.52	2.56	2.60	5.26	2.94	2.18	5.03	3.50	1.96
		30	3.57	2.33	1.37	4.11	2.31	1.68	3.56	1.97	1.70
b	$(0, \gamma, \gamma, \gamma)$ **50**	**5**	3.83	2.14	2.02	4.05	1.73	1.60	3.66	1.85	2.26
		10	3.24	1.76	1.70	3.40	1.81	1.83	3.22	1.74	1.55
		20	2.11	1.34	1.16	2.25	1.45	1.21	2.03	1.69	0.90
		30	1.30	0.77	0.86	1.58	0.89	0.64	1.86	0.77	0.46
c	$(0, \gamma, 2\gamma, 3\gamma)$ **25**	**5**	3.61	3.04	1.41	4.24	1.55	2.15	4.84	2.29	2.45
		10	3.30	1.97	1.98	3.45	2.05	2.05	4.15	1.63	1.74
		20	2.24	1.53	0.65	2.32	1.04	0.91	2.96	1.45	1.02
		30	1.81	0.90	0.57	1.37	1.01	0.52	1.82	1.27	0.69
d	$(0, \gamma, \gamma, \gamma)$ **25**	**5**	1.66	1.00	1.03	1.63	0.80	1.02	1.50	0.86	0.96
		10	0.90	0.62	0.61	0.88	0.86	0.77	1.50	0.84	0.71
		20	1.10	0.63	0.39	0.88	0.44	0.33	0.93	0.58	0.37
		30	0.73	0.58	0.44	0.79	0.57	0.31	0.48	0.45	0.45

Table 2: Percentage difference in total cost between the DA solution and the optimal solution.

respectively.

2.5 Prohibited turns

In many real-world situations, some turns are not just more expensive, they are prohibited. For example, a truck cannot make a U-turn on a narrow city street. Taking prohibited turns into account in a direct approach can be important. In a solution approach using a graph transformation, it is straightforward to take into account the prohibited turns. To transform the arc routing problem into an equivalent node routing problem, it is sufficient to remove the arcs related to the prohibited turns from the new graph.

For the DA, it is not straightforward. In the first stage, the graph G is transformed into an Eulerian graph G_e. This transformation does not take the prohibited turns into account. In the second stage, an end-pairing algorithm is used to generate an Eulerian tour. The second stage uses the turn penalties. One way to take into account the prohibited turns is to allow all turns in the Eulerian graph G_e (to ensure a feasible DRPP tour), but associate a big cost M to the prohibited turns in the transformed graph. Unfortunately, the first stage may result in an Eulerian graph that would force one or more prohibited turns into the final solution.

Here is an alternative way to consider the prohibited turns. Suppose the path $i - j - k$ from vertex i to vertex k uses the arcs (i, j) and (j, k). If the path $i - j - k$ corresponds to a prohibited turn, we can remove arcs (i, j) and (j, k) from the solution and replace them with the shortest path from i to k that avoids the prohibited turns [22]. We note that, if one or both arcs (i, j) and (j, k) are required, the resulting new tour could be infeasible.

Number of Nodes			8 × 8			10 × 10			12 × 12		
	Required Arcs (%)		25	50	75	25	50	75	25	50	75
	Removed Arcs (%)										
a	$(0, \gamma, 2\gamma, 3\gamma)$	5	-5.70	-5.34	-3.40	-6.90	-5.12	-3.12	-7.84	-4.71	-3.66
		10	-6.02	-3.07	-2.01	-6.17	-3.24	-1.84	-6.96	-4.02	-2.64
	50	20	-5.06	-1.94	-1.00	-5.12	-2.93	-1.59	-5.96	-2.26	-1.33
		30	-2.48	-1.56	-1.09	-3.36	-1.72	-0.70	-3.18	-1.96	-1.00
b	$(0, \gamma, \gamma, \gamma)$	5	-2.27	-0.65	-0.64	-2.72	-0.53	-0.80	-1.81	-0.74	-0.76
		10	-1.44	-0.53	-0.59	-2.13	-0.63	-0.53	-2.07	-0.80	-0.50
	50	20	-0.82	-0.58	-0.56	-1.19	-0.55	-0.30	-0.84	-0.64	-0.34
		30	-0.33	-0.20	-0.19	-0.52	-0.42	-0.36	-0.71	-0.27	-0.23
c	$(0, \gamma, 2\gamma, 3\gamma)$	5	-2.47	-1.01	-0.46	-2.79	-0.74	-0.58	-2.06	-0.89	-0.61
		10	-1.35	-0.56	-0.49	-1.79	-0.74	-0.63	-2.44	-0.71	-0.59
	25	20	-1.12	-0.43	-0.27	-1.30	-0.58	-0.23	-1.02	-0.67	-0.54
		30	-0.59	-0.19	-0.19	-0.87	-0.37	-0.48	-0.65	-0.46	-0.28
d	$(0, \gamma, \gamma, \gamma)$	5	-0.71	-0.45	-0.39	-0.93	-0.39	-0.40	-0.99	-0.38	-0.41
		10	-0.58	-0.23	-0.37	-0.59	-0.43	-0.33	-0.65	-0.36	-0.31
	25	20	-0.54	-0.22	-0.15	-0.39	-0.28	-0.21	-0.47	-0.34	-0.14
		30	-0.25	-0.28	-0.18	-0.30	-0.21	-0.15	-0.29	-0.23	-0.21

Table 3: Percentage difference in tour length between the DA solution and the optimal solution.

Number of Nodes			8 × 8			10 × 10			12 × 12		
	Required Arcs (%)		25	50	75	25	50	75	25	50	75
	Removed Arcs (%)										
a	$(0, \gamma, 2\gamma, 3\gamma)$	5	34.71	46.83	28.72	43.60	45.95	42.13	56.00	53.00	43.91
		10	36.43	25.92	17.17	40.02	28.70	20.65	46.59	35.00	25.67
	50	20	26.32	13.08	11.68	26.47	17.95	12.34	28.46	18.15	10.79
		30	14.88	10.36	7.07	18.44	11.09	7.01	16.65	10.55	7.82
b	$(0, \gamma, \gamma, \gamma)$	5	24.82	15.57	17.67	28.50	14.30	16.01	22.51	16.76	22.51
		10	19.18	11.99	13.57	23.19	14.26	14.91	22.28	14.76	13.10
	50	20	11.47	8.92	8.72	13.82	9.64	7.72	11.63	11.64	6.49
		30	6.58	4.28	4.74	8.39	5.82	4.51	10.37	4.59	3.09
c	$(0, \gamma, 2\gamma, 3\gamma)$	5	27.11	27.23	13.21	33.31	14.81	22.10	32.89	23.71	26.50
		10	19.87	14.33	17.69	23.29	17.29	19.39	31.32	14.73	17.68
	25	20	14.03	9.49	5.11	15.49	8.33	6.69	17.77	11.30	9.12
		30	10.28	5.09	3.79	9.17	6.43	4.94	10.85	8.40	5.05
d	$(0, \gamma, \gamma, \gamma)$	5	17.01	14.55	16.76	17.97	13.02	17.94	17.66	15.07	18.90
		10	9.97	7.91	11.01	10.33	13.27	12.04	16.28	12.74	12.32
	25	20	11.99	7.50	4.73	9.09	6.20	4.99	9.96	8.33	4.89
		30	6.84	6.73	5.04	7.59	6.25	3.79	5.30	5.45	5.51

Table 4: Percentage difference in turn costs between the DA solution and the optimal solution.

Required Arcs (%) / Removed Arcs (%)	**25**	**50**	**75**	Average
5	12.9	4.9	0.6	6.1
10	13.5	8.7	2.7	8.3
20	17.4	7.4	3.8	9.5
30	15.4	9.9	6.2	10.5
Average	14.8	7.7	3.3	8.6

Table 5: Percentage of prohibited turns in the DA solution.

To investigate how likely it is that a DA solution has prohibited turns, we perform the following experiment. We run ILP on the directed graph to create the Eulerian graph G_e described in Section 2.1. Next, we run α-ILP with $\alpha = 0$ on its transformed graph G' with all arcs required as input. To avoid the prohibited turns, we set the cost of the allowed turns to zero and the cost of the prohibited turns to one. The size of the input graph is fixed at 12×12.

The results of this experiment are given in Table 5. We see that the number of prohibited turns in a DA solution is related to the density of the graph and the number of required arcs. As the number of required arcs or the density of the graph increases, the percentage of prohibited turns decreases. For example, when 75% of the arcs are required and we remove only 5% of the total number of arcs, the percentage of prohibited turns decreases to 0.6%. The formula used to compute the values reported in Table 5 is $100 \times \frac{Ft(DA[ILP,\alpha\text{-ILP}])}{Tt(OPT)}$. Unfortunately, the results of this experiment show that DA is not guaranteed to produce solutions that are free of prohibited turns. If there exists a solution for the DRPP-TP without prohibited turns, it is not guaranteed that DA can produce such a solution. For this reason, DA may not be applicable to problems with prohibited turns.

3 Penalty ILP model

In this section, we focus on the first stage of DA. Our penalty variant (denoted by β-ILP) extends the ILP model for the DRPP and produces an Eulerian graph covering all required arcs. The output graph takes into account the turn penalties. In particular, for each arc $a = (i, j) \in A$, we define $d(a) = d(i, j) = (j, k) \in A$ such that the path (i, j, k) produces the minimum turn cost after we traverse arc (i, j).

We define the following variables and constant:

$y_a \in \mathbb{N}_{\geq 0}$ is the number of times the solution traverses arc $a \in A$ but does not traverse arc $d(a)$.

β is a penalty value.

Our penalty model is presented below:

$$\text{(β-ILP)} \qquad \text{Minimize } \sum_{a\in A} c(a)x_a + \beta \sum_{a\in A} y_a \tag{11}$$

$$\text{Constraints } (2)-(9)$$

$$y_a \geq x_a - x_{d(a)} \qquad \forall\ a \in A \tag{12}$$

$$y_a \in \mathbb{N}_{\geq 0} \qquad \forall\ a \in A. \tag{13}$$

Using constraint (12), the variable y_a counts the number of times that arc a is traversed and arc $d(a)$ is not traversed. We define $(\gamma_1(a), \gamma_2(a), \gamma_3(a), \ldots)$ as the costs (in ascending order) associated with the different turns that we can make after traversing arc a. We define $\gamma(a) = \gamma_2(a) - \gamma_1(a)$ as the minimum opportunity cost if we do not select the best turn in terms of cost after traversing arc a.

Theorem 3.1 *In the Eulerian graph produced by β-ILP model with* $\beta = \min_{a\in A}\gamma(a)$*, the term* $(\sum_{a\in A}\gamma_1(a)x_a + \beta\sum_{a\in A} y_a)$ *gives a lower bound on the total cost of turns.*

Proof In the Eulerian graph, we have a tour that traverses each arc one time. After traversing each arc a, we have a turn and $\gamma_1(a)$ represents the minimum cost of this turn. In the lower bound, $\sum_{a\in A}\gamma_1(a)x_a$ represents the sum of all minimum costs related to the turns. The variable y_a counts the number of times that it is not possible to select the best turn after traversing arc a. $\beta\sum_{a\in A} y_a$ is the cost related to each time that it is not possible to select the best turn after traversing an arc. Because $\beta = \min_{a\in A}\gamma(a)$, the term $(\sum_{a\in A}\gamma_1(a)x_a + \beta\sum_{a\in A} y_a)$ gives a lower bound on the total cost of turning.

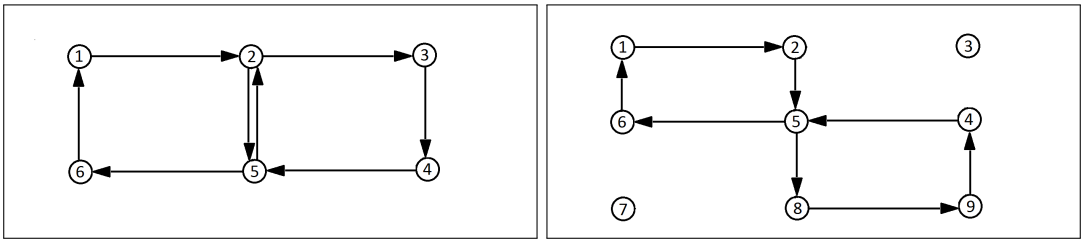
4 Node-to-node approach

In this section, we focus on the second stage of DA. We assume that an Eulerian graph is given. The tour length is fixed for all Eulerian tours, but the turn cost depends on the sequence in which the arcs are traversed. Based on the observation that each turn cost is incurred at a visit to a node, we estimate the turn cost of a tour by considering different possible turns made at every node. The three procedures described in the subsections (lower bound, greedy, local search algorithm) are developed from this observation.

4.1 Lower bound on the turn cost

Proposition 4.1 *In an Eulerian tour, the sum of the minimum possible turn costs at each node is a lower bound on the turn costs of that tour, no matter the order in which arcs are traversed.*

In this subsection, we assume a turn cost structure of $(0, 1, 2, 3)$ for illustration. In Fig. 3(a), we have an Eulerian graph and want to determine a lower bound on the turn cost. It is only possible to make a right turn at node 1, so the minimum turn cost incurred at node 1 is 1. Similarly, the minimum turn cost is 1 at nodes 3, 4, and 6. There are two incoming arcs and two outgoing arcs at node 2, so there are two possible turn combinations, either two right turns ($1-2-5$ and $5-2-3$) with a total cost of 2 or one straight turn ($1-2-3$) and one U-turn ($5-2-5$) with a total cost of 3. The minimum value for these two possibilities is 2. It follows that a lower bound



(a) (b)

Figure 3: Eulerian graph with possible turns.

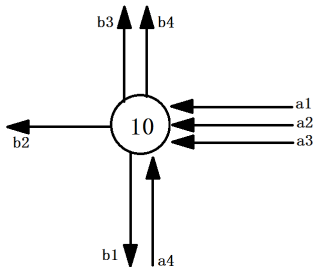


Figure 4: Turns at a node.

on the turn cost at node 5 is also 2. Hence, a lower bound on the turn cost for this Eulerian graph is 8. The actual minimum turn cost for this small graph is 9 by inspection, using the tour $(1-2-3-4-5-2-5-6-1)$. We point out that there is no depot, or starting node, specified for the tour. Therefore, a turn cost of 1 is incurred at node 1. The lower bound obtained with this procedure may be tight. For example, in Fig. 3(b), the minimum turn cost at node 5 is 0 with two straight turns ($2-5-8$ and $4-5-6$). There is only one possible turn at the other nodes. A lower bound is 9 and this bound is attainable using the tour $(1-2-5-8-9-4-5-6-1)$.

In our implementation, instead of enumerating all possible turn combinations at a node, we solve an assignment problem and apply the Hungarian algorithm [15] to assign the outgoing arcs to the incoming arcs. The assignment problem is applicable because the number of incoming arcs is equal to the number of outgoing arcs at every node in an Eulerian graph. The assignment cost is the same as the turn cost. For example, in Fig. 4, node 10 has four incoming arcs $a1$ to $a4$, and four outgoing arcs $b1$ to $b4$. If arc $b1$ is assigned to arc $a4$, a U-turn results, so the cost is 3. Other assignment costs are presented in Table 6. A similar idea was used by Bodin and Kursh [4], but the assignment problem was solved on a node that was not necessarily in an Eulerian graph.

4.2 Greedy algorithm

We propose a greedy algorithm (denoted by GRA) that extends the lower bound to obtain a high-quality solution with respect to the turn cost. For each node in the Eulerian graph G_e, we solve the assignment problem that produces an arrangement of the turns with minimum cost. If this arrangement does not disconnect the graph (in other words, if this arrangement does not create two or more disjoint loops (see Fig. 5(b))), it is adopted. Otherwise, if the number of incoming arcs at the node is less than a specified value (we denote this parameter by dn, which is equal to seven in our experiments), we enumerate all possible assignments and select the best one in terms of lowest turn cost without disconnecting the graph. If there are more than dn incoming arcs, we pair the incoming arcs one by one with the outgoing arcs in the least-cost manner without

	a1	a2	a3	a4
b1	2	2	2	3
b2	0	0	0	2
b3	1	1	1	0
b4	1	1	1	0

Table 6: Illustration of assignment costs, using the network of Figure 4.

Let:
$G = (N, A)$ be a directed Eulerian graph
$S \in \mathcal{P}(N)$, where $\mathcal{P}(N)$ is the set of permutations on N

1 $U = \emptyset$ is a set of turns
2 For each $n \in S$
3 Let the set of turns $T = \emptyset$
4 While number of unpaired incoming arcs at $n > dn$
Pair an unpaired incoming arc with an unpaired outgoing arc that results in a least cost turn, t
$T = T \cup \{t\}$
5 Identify a set of turns, T', at n for the unpaired outgoing arcs to the unparied incoming arcs,
with minimum cost by solving an assignment problem
6 If Connect(G,$U \cup T \cup T'$) = false
7 Enumerate all possible assignments of the unpaired outgoing–incoming arcs, and
choose the least cost assignment that results in a set of turns, T', such that Connect(G,$U \cup T \cup T'$) = true
8 $U = U \cup T \cup T'$
9 Return U

Table 7: Greedy algorithm (GRA).

disconnecting the graph until there are dn incoming arcs that are not paired. Then we enumerate all possible assignments of the dn incoming arcs. After we have selected the best arrangement for each node without disconnecting the graph, we generate a feasible solution to the Eulerian graph. To illustrate, in Fig. 3, there is only one possible arrangement at nodes $1, 3, 4$, and 6. At node 2, the least-cost arrangement has two right turns and this arrangement does not disconnect the graph (Fig. 5(a)). Similarly, at node 5, the least-cost arrangement has two right turns. However, given the arrangement at node 2, the same arrangement at node 5 will disconnect the graph into two pieces, one with nodes 1 and 6 and one with nodes 3 and 4 (Fig. 5(b)). Therefore, we can have only one U-turn and one straight turn at node 5 (Fig. 6(a)). The tour that results from the greedy algorithm is $(1-2-5-2-3-4-5-6-1)$ with a turn cost of 9. The greedy algorithm and the algorithm to test graph connectivity are given in Tables 7 and 8, respectively.

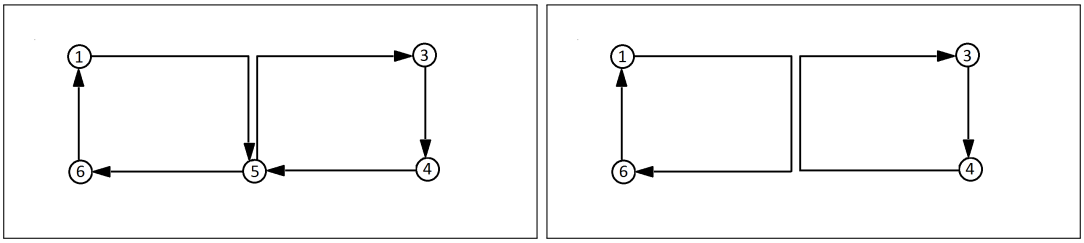
4.3 Local search algorithm

Our greedy algorithm depends on the order in which the nodes are considered. For example, if node 5 is considered before node 2 in Fig. 3(a), we produce a different tour $1-2-3-4-5-2-5-6-1$

Let:
$G = (N, A)$ be a directed Eulerian graph
$U \subseteq \{(i, j, k) | i, j, k \in N \text{ and } (i, j)(j, k) \in A\}$

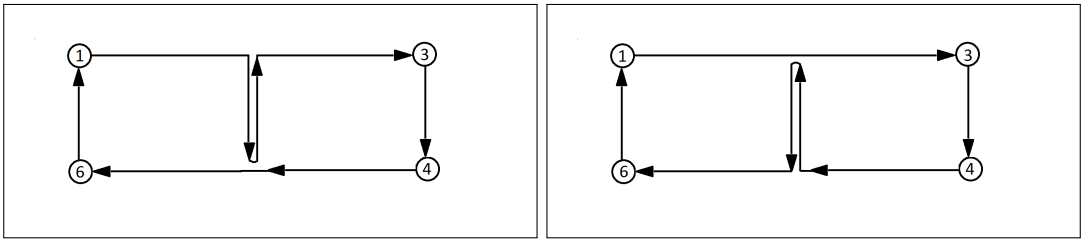
1 x = number of nodes visited by a Depth-First Search (DFS) of $G(N, A)$
starting from a random node.
If DFS uses the arc (i, j) and $(i, j, k) \in U$,
then to account for the turn (i, j, k), DFS is forced to use the arc (j, k).
2 Return true if $x = |N|$

Table 8: Connect algorithm.



(a) (b)

Figure 5: Disconnected graph.



(a) (b)

Figure 6: U-turns.

(shown in Fig. 6(b)). Therefore, we use a local search algorithm (LSA) to improve the solution produced by the greedy algorithm. Given a greedy solution, we randomly permutate 5% of the node positions in the sequence and apply the same procedure in the greedy algorithm to produce a feasible solution. If this solution is better than the current solution, we retain it. Our algorithm is given in Table 9.

4.4 Results of computational experiments

In this section, we report on the results produced by GRA and LSA. Since both strategies are used in the second stage of DA, their results are compared to the results produced by strategy $v4$ given in [9] (taking into account that strategy $v4$ outperformed all the other strategies proposed in [9]), the optimal solution produced by α-ILP with $\alpha = 0$ (denoted by OPT), and the algorithm developed by Edmonds and Johnson [13] (denoted by EJ). We use the input graph defined in Section 2.4.1. An input graph is transformed to an Eulerian graph using the ILP model. The test

	Let:
	$G = (N, A)$ be a directed Eulerian graph
	$S_t \in \mathcal{P}(N) : \lvert S_t \rvert = \lvert N \rvert$, S_t is a permutation of N
	$\mu(S_t)$ = solution value of the greedy algorithm with input S_t and $G(N, A)$
	$\epsilon(S_t)$ = neighborhood of S_t (in which p percentage of the nodes differ from S_t)
	l = size of our neighborhood
1	Set index $t = 0$, $p = 5\%$, $l = 10$
2	Construct a random permutation S_0
3	Do
4	Create $P = \{S \in \epsilon(S_t)\}$ containing l random permutations $S \in \epsilon(S_t)$
5	Choose $S' \in P$ such that $\mu(S') \leq \mu(S)$ $\forall S \in P$
6	Set $S_{t+1} = S'$
7	Set $t = t + 1$
8	While $\mu(S_t) \leq \mu(S_{t-1})$
9	Return S_{t-1}

Table 9: Local search algorithm (LSA).

scenarios are the same as defined in Section 2.4.2.

For each scenario of Table 10, we use 10 12×12 Eulerian graphs. In order to compute the values shown in this table, we use the formula $100 \times \frac{V(DA[ILP,X])-V(OPT)}{V(OPT)}$, where X is EJ, $v4$, GRA and, LSA, respectively.

In the first part of Table 10, for a straight turn, right turn, left turn, and U-turn, we have costs of $(0, 1, 2, 3)$, respectively. Here, the number of turns is the average number of edges in the Eulerian graph. We see that strategy $v4$ is effective, because it decreases the turn costs given by the EJ solution. The average gap of $v4$ with respect to the optimal value is 49.5%. GRA outperforms $v4$ with an average gap of 4.5% compared to the optimal value. LSA produces better solutions, with an average gap equal to 1.4%.

In the second part of Table 10, for a straight turn, right turn, left turn, and U-turn, we have costs of $(0, 1, 1, 1)$, respectively. In this scenario, the average gap between $v4$ and the optimal value is 25.4%. GRA produces an average gap of 2.0%. LSA produces an average gap of 0.8%. The running times are given in Table 11. On average, the running time of GRA is about twice that of $v4$. LSA takes longer than $v4$ and GRA, but it is still much faster than OPT.

In Tables 12 and 13, we compare the total cost. In Table 12, we compare the DA using the optimal solution produced by α-ILP in the second stage with the DA using LSA in the second stage. The table entries represent the percentage differences between the two approaches. These percentage differences are calculated using the formula $100 \times \frac{V(DA[ILP,LSA])-V(DA[ILP,\alpha\text{-ILP}])}{V(DA[ILP,\alpha\text{-ILP}])}$. In the subtables **b** and **d**, we have, for a straight turn, right turn, left turn, and U-turn, the cost $(0, \gamma, \gamma, \gamma)$ (with γ equal to 50 in **b** and 25 in **d**, respectively). Each row gives the percentage of arcs that we remove from a complete grid representing our input graph. Each column gives the percentage of arcs that we need to visit in the tour. In the subtables **a** and **c**, we have, for a straight turn, right turn, left turn, and U-turn, the cost $(0, \gamma, 2\gamma, 3\gamma)$ (with γ equal to 50 in **a** and 25 in **c**, respectively). For example, in subtable **a** of Table 12, when the number of nodes is 8×8, 5% of the arcs are removed, and 50% of the arcs are required, there is an increase in total cost of 0.13%, on average, if the second stage of the exact end-pairing procedure in DA is replaced by LSA. We see that the additional cost (beyond the exact end-pairing solution) from using LSA is less than or equal to 0.31% in every case.

In Table 13, we compare the optimal solution for the DRPP-TP produced by α-ILP to the DA solution based on β-ILP and LSA. We set $\alpha = 0.5$ for α-ILP and $\beta = \gamma \in \{25, 50\}$ for β-ILP. The percentage differences in solutions are calculated using the formula $100 \times \frac{V(DA[\beta\text{-ILP},LSA])-V(OPT)}{V(OPT)}$. The results show that DA is more effective if the first stage takes into account the turn penalties. For example, in subtable **a** of Table 13, when the number of nodes is 8×8, 5% of the arcs are removed, and 25% of the arcs are required, the gap between our heuristic solution using β-ILP and LSA and the optimal DRPP-TP solution is 3.59%, on average. The gap between the DA solution and the optimal solution in this scenario is 6.86%, on average (see Table 2). The average gap between the optimal solution and DA is equal to $(1.65\%, 0.56\%, 0.63\%, 0.23\%)$ for subtables ($\mathbf{a}, \mathbf{b}, \mathbf{c}$, and $\mathbf{d}$). Comparing these results to the same average results of Table 2 $(4.32\%, 1.85\%, 2.00\%, 0.79\%)$, we see that accounting for turn costs in the first stage of DA produces high-quality solutions.

In Table 14, we compare the DA solution, based on β-ILP and LSA, with the algorithms proposed by Benavent and Soler [2] and Bodin et al. [3]. Benavent and Soler proposed three

heuristics for the DRPP-TP. Two of them are constructive, whereas the third is an improvement heuristic. Their best results combined the second constructive heuristic with the improvement approach (CONS2+IMP). Bodin et al. [3] proposed a constructive approach (BK) that does not take into account prohibited turns. We performed our experiments on the instances used in [2]. Each instance is characterized by a specific cost for turns and a large number of prohibited turns. In our computational experiments, we associate a large cost of M with each prohibited turn. We apply our approach 10 times using the values (30, 60, 90, ..., 300) for β. In Tables 14 and 15, we display the best result over 10 runs of the algorithm for each instance. The running time is the total for the 10 iterations. For the set of instances in Table 14, the running time for ItLSA is negligible. The average running times over all instances are equal to (0.4, 15.43, 7.59) seconds for (ItLSA, CONS2+IMP, BK), respectively and the maximum running times are (1.8, 90.43, 29.51) seconds, respectively. It is important to point out that the experiments for CONS2+IMP and BK are performed on a Sun Sparcstation 20.

In 12 cases, ItLSA produces the best results. In 13 cases, CONS2+IMP produces the best results and BK never produces a best result. Even though the instances contain numerous prohibited turns, our approach (ItLSA) is capable of producing very effective solutions.

In Table 15, we use real street networks to evaluate the effectiveness of the approach. The numbers of vertices, arcs, and required arcs are shown for each instance (see Capobianco et al. [6] for more details). We used ∝-ILP to compute the optimal solution. Unfortunately, none of the instances were solved optimally in three hours of running time. In the biggest instance, we were unable to produce a feasible solution. On the other hand, using β-ILP and LSA, we were able to produce feasible solutions for all instances. The average gap with respect to the best solution was less than 1.2 percent.

5 Conclusions

The most common solution approach for the directed rural postman problem with turn penalties involves transforming an arc routing problem into an equivalent node routing problem. In this paper, we investigated the applicability of a direct approach. The direct approach proposed in the literature has two stages. The first stage focuses on the directed rural postman problem. The second stage accounts for the turn penalties. We designed computational experiments to identify the features of the input graph that make the direct approach effective. In particular, taking into account the costs of the turns, the density of the graph, and the percentage of required arcs, the cost of the solution produced by the direct approach was only 0.4% to 0.9% greater than the cost of the optimal solution.

We also tested DA when turns are prohibited. In our tests, all parameter settings produced solutions with prohibited turns, so DA is not recommended in this case.

In the second part of this paper, we developed a greedy algorithm and a local search procedure to optimize the turn cost in an Eulerian graph. We compared our results to the results of algorithms proposed in the literature and observed that our algorithm produced better solutions. Finally, we proposed a mathematical model for the directed rural postman problem that accounted for turn costs. Combining this mathematical model with our local search algorithm produced the best solutions. We tested this new approach on instances from the literature and on real street

networks. In both cases, we were able to produce effective results in reasonable running times, even taking into account that the instances from the literature contain numerous prohibited turns.

In future work, we want to investigate the applicability of DA when prohibited turns are considered in the first phase. We would like to improve the heuristic approach using an iterated greedy [19] or a carousel greedy algorithm [7]. Both approaches would allow us to improve the results without greatly increasing the running times. Finally, we might consider replacing local search with a more sophisticated metaheuristic.

	Removed Arcs (%)		Number of	Cost				
					Increased Cost over OPT (%)			
	Required Arcs (%)		Turns	**OPT**	**EJ**	**v4**	**GRA**	**LSA**
Turn Costs $(0, 1, 2, 3)$	**25**	**5**	222.0	293.5	13.7	9.9	3.7	0.9
		10	219.2	289.4	19.2	10.5	3.8	0.6
		20	211.6	264.3	19.3	10.1	1.0	0.0
		30	218.0	271.7	19.9	8.6	1.6	0.0
	50	**5**	356.8	272.1	191.1	76.9	4.6	1.2
		10	356.6	308.6	188.9	58.6	7.0	1.4
		20	351.0	347.6	145.5	53.1	6.2	2.7
		30	346.2	364.8	111.6	49.0	5.0	0.5
	75	**5**	474.6	300.1	388.6	86.9	6.7	4.2
		10	466.6	323.5	342.6	93.0	5.6	3.1
		20	469.6	394.4	281.7	73.4	5.2	1.4
		30	473.6	441.2	220.5	64.0	3.9	1.1
	Average				161.9	49.5	4.5	1.4
Turn Costs $(0, 1, 1, 1)$	**25**	**5**	220.0	154.9	9.2	5.5	1.4	0.3
		10	216.2	154.0	9.9	5.8	1.4	0.3
		20	212.2	140.3	8.9	5.5	1.0	0.0
		30	214.8	146.2	11.2	4.6	0.7	0.3
	50	**5**	357.2	142.5	94.8	34.5	4.5	1.9
		10	353.8	152.9	84.4	32.7	3.9	1.3
		20	352.0	186.9	66.7	26.5	3.0	1.0
		30	372.2	217.7	46.0	17.9	1.4	0.6
	75	**5**	472.4	180.5	169.3	46.9	2.3	1.0
		10	463.0	182.1	158.4	47.9	1.7	1.4
		20	471.4	228.7	126.7	41.6	1.8	1.2
		30	481.4	255.6	90.8	34.9	1.2	0.5
	Average				73.0	25.4	2.0	0.8

Table 10: Turn cost comparison of EJ, $v4$, GRA, and LSA.

	Removed Arcs (%)		Time (milliseconds)			
	Required Arcs (%)		**OPT**	**v4**	**GRA**	**LSA**
Turn Costs $(0, 1, 2, 3)$	**25**	**5**	194.1	0.5	0.7	12.5
		10	167.3	0.3	0.8	11.2
		20	129.6	0.3	0.4	8.0
		30	216.1	0.1	0.9	9.1
	50	**5**	1081.1	0.9	1.6	19.5
		10	1052.7	0.8	1.1	21.1
		20	1357.8	0.8	1.1	16.7
		30	1090.2	0.6	1.5	17.4
	75	**5**	3090.0	0.8	2.0	23.8
		10	1620.6	0.9	1.4	25.8
		20	2542.2	1.2	1.7	23.3
		30	2846.7	1.1	1.9	19.5
Turn Costs $(0, 1, 1, 1)$	**25**	**5**	109.1	0.2	0.9	8.9
		10	133.9	0.3	0.6	8.6
		20	136.1	0.3	0.8	7.2
		30	173.5	0.5	0.5	7.0
	50	**5**	1155.5	0.6	1.1	15.5
		10	1075.1	0.6	1.0	17.2
		20	948.3	1.0	1.2	13.9
		30	1273.7	0.6	1.3	13.1
	75	**5**	2005.6	1.0	1.6	23.2
		10	1710.0	0.9	1.4	16.2
		20	2290.0	1.1	1.5	17.7
		30	2264.0	1.0	1.6	15.9

Table 11: Average running times of OPT, $v4$, GRA, LSA for the experiments shown in Table 10.

Number of Nodes			8 × 8			10 × 10			12 × 12		
	Required Arcs (%)		25	50	75	25	50	75	25	50	75
	Removed Arcs (%)										
a	$(0, \gamma, 2\gamma, 3\gamma)$	**5**	0.00	0.13	0.17	0.00	0.20	0.31	0.05	0.17	0.16
		10	0.00	0.18	0.30	0.06	0.22	0.16	0.01	0.09	0.06
	50	**20**	0.10	0.02	0.10	0.02	0.07	0.09	0.00	0.05	0.10
		30	0.00	0.00	0.02	0.00	0.15	0.04	0.01	0.03	0.05
b	$(0, \gamma, \gamma, \gamma)$	**5**	0.00	0.08	0.15	0.02	0.18	0.08	0.02	0.22	0.11
		10	0.00	0.13	0.08	0.00	0.13	0.10	0.00	0.15	0.07
	50	**20**	0.00	0.03	0.08	0.00	0.18	0.04	0.04	0.09	0.04
		30	0.00	0.13	0.02	0.00	0.03	0.06	0.05	0.04	0.02
c	$(0, \gamma, 2\gamma, 3\gamma)$	**5**	0.00	0.07	0.10	0.00	0.09	0.08	0.05	0.15	0.15
		10	0.00	0.05	0.23	0.05	0.06	0.07	0.07	0.09	0.09
	25	**20**	0.00	0.05	0.05	0.01	0.05	0.05	0.02	0.08	0.08
		30	0.00	0.00	0.00	0.00	0.03	0.06	0.00	0.03	0.02
d	$(0, \gamma, \gamma, \gamma)$	**5**	0.02	0.07	0.01	0.00	0.08	0.06	0.00	0.14	0.03
		10	0.02	0.07	0.02	0.01	0.08	0.07	0.00	0.10	0.06
	25	**20**	0.00	0.00	0.02	0.00	0.05	0.03	0.01	0.06	0.04
		30	0.00	0.03	0.03	0.00	0.02	0.03	0.01	0.02	0.05

Table 12: Percentage difference in total cost between two DA solutions using α-ILP and LSA, respectively, in the second stage.

Number of Nodes			8 × 8			10 × 10			12 × 12		
	Required Arcs (%)		25	50	75	25	50	75	25	50	75
	Removed Arcs (%)										
a	$(0, \gamma, 2\gamma, 3\gamma)$	**5**	3.59	1.87	1.06	4.07	2.35	0.98	3.72	1.89	1.31
		10	3.59	1.41	0.61	3.61	1.79	0.97	3.59	1.88	0.82
	50	**20**	1.77	0.93	0.53	2.43	1.29	0.75	2.69	1.25	0.65
		30	1.28	0.74	0.44	1.16	0.93	0.35	1.31	0.93	0.78
b	$(0, \gamma, \gamma, \gamma)$	**5**	1.26	0.76	0.36	0.94	0.65	0.48	0.97	0.60	0.36
		10	0.92	0.86	0.18	1.42	0.54	0.23	1.06	0.57	0.31
	50	**20**	0.96	0.41	0.20	0.89	0.41	0.21	0.72	0.43	0.25
		30	0.51	0.37	0.09	0.58	0.40	0.07	0.58	0.35	0.22
c	$(0, \gamma, 2\gamma, 3\gamma)$	**5**	1.69	0.97	0.31	1.74	0.79	0.32	1.61	0.69	0.30
		10	1.07	0.87	0.46	1.05	0.58	0.40	1.24	0.59	0.34
	25	**20**	0.87	0.44	0.22	0.84	0.40	0.25	0.96	0.46	0.20
		30	0.39	0.21	0.21	0.50	0.31	0.36	0.42	0.44	0.17
d	$(0, \gamma, \gamma, \gamma)$	**5**	0.52	0.32	0.16	0.39	0.55	0.16	0.42	0.26	0.11
		10	0.19	0.19	0.08	0.33	0.33	0.11	0.40	0.32	0.18
	25	**20**	0.21	0.20	0.10	0.33	0.17	0.10	0.31	0.21	0.12
		30	0.25	0.10	0.09	0.32	0.13	0.05	0.21	0.14	0.09

Table 13: Percentage difference in total cost between β-ILP+LSA solution and the optimal solution.

	$\|V\|$	$\|A\|$	$\|A_r\|$	Turns		Penalty	Cost				
				All	Proh.		**BK**		**CONS2+IMP**	**ItLSA**	
P1	35	116	116	400	50	(1,2,4,8)	21147		21137	**21116**	
P2	36	120	79	416	90	(1,3,5,11)	13936		**13916**	13918	
P3	40	132	98	452	73	(1,3,5,11)	15634		15636	**15620**	
P4	49	168	148	596	129	(1,1,3,7)	16512		16418	**16382**	
P5	49	168	128	596	135	(1,2,3,7)	22011	+4M	22227	**22214**	
P6	54	186	166	662	86	(1,2,4,8)	26959		**26950**	**26950**	
P7	60	208	131	744	86	(2,2,4,8)	24174	+4M	**24546**	24394	+M
P8	96	344	212	1264	234	(0,2,4,8)	19792	+8M	**20574**	20334	+3M
P9	96	344	344	1264	259	(0,2,4,8)	36972		36944	**36912**	
P10	99	356	176	1312	384	(1,2,3,M)	17374	+14M	**18135**	18623	+3M
P11	100	360	360	1328	326	(1,3,5,11)	37078		37088	**37028**	
P12	105	376	376	1380	217	(2,1,4,8)	35046		34815	**34613**	
P13	105	376	177	1380	396	(1,1,4,8)	24770	+28M	**26100**	27021	+5M
P14	126	458	354	1702	231	(2,3,5,11)	25686	+M	35623	**35620**	
P15	126	458	458	1702	469	(2,3,5,11)	36441		36449	**36398**	
P16	143	524	358	1960	522	(2,3,5,11)	32207	+6M	**32766**	33062	
P17	143	524	524	1960	517	(2,3,5,11)	42369		42363	**42308**	
P18	144	528	377	1976	413	(2,1,4,8)	53565	+5M	**36036**	36991	
P19	144	528	427	1976	311	(2,1,4,8)	40886	+3M	**41150**	41389	+M
P20	150	550	425	2058	367	(1,3,5,11)	38339	+3M	**38564**	38423	+M
P21	150	550	550	2058	360	(1,3,5,11)	40338		40338	**40260**	
P22	169	624	556	2348	650	(1,1,3,8)	43990	+5M	**44176**	45050	
P23	169	624	451	2348	584	(1,1,3,8)	34777	+12M	**35224**	36564	
P24	180	666	540	2510	754	(1,2,5,M)	48312	+3M	**48361**	49038	
Number of best solutions							**0**		**13**	**12**	

Table 14: Iterative β-ILP and LSA (ItLSA) compared to CONS2+IMP [2] and BK [3].

	$\|V\|$	$\|A\|$	$\|A_r\|$	**OPT**		**β-ILP + LSA**	
				Solution	Time1 (s)	Solution	Time2 (s)
Campobasso	3489	7176	1794	371797*	1412	375770	48
Salerno	3811	7602	1901	489284*	1027	491007	51
Boston	16407	33561	8391	1833241*	3067	1856919	496
Rome	18585	36894	9224	1856639*	5280	1888187	828
Washington, DC	27766	61743	15436	–	–	3295652	4831

Table 15: Comparison of the best solution found (α-ILP with α= 0.5) with our heuristic solution on real street networks where * indicates a suboptimal solution and – indicates no feasible solution found within three hours. The column Time1 shows the number of seconds necessary for α-ILP with α= 0.5 to produce a solution as good or better than our heuristic solution. The column Time2 shows the running times in seconds of our heuristic approach. We used the penalties (0, 25, 50, 75).

References

[1] I. Ari, V. Aksakalli, V. Aydogdu, and S. Kum. Optimal ship navigation with safety distance and realistic turn constraints. *European Journal of Operational Research*, 229(3):707–717, 2013.

[2] E. Benavent and D. Soler. The directed rural postman problem with turn penalties. *Transportation Science*, 33(4):408–418, 1999.

[3] L. Bodin, G. Fagin, R. Welebny, and J. Greenberg. The Design of a Computerized Sanitation Vehicle Routing and Scheduling System for the Town of Oyster Bay, New York. *Computers & Operations Research*, 16:45–54, 1989.

[4] L. Bodin and S. Kursh. A computer-assisted system for the routing and scheduling of street sweepers. *Operations Research*, 26(4):525–537, 1978.

[5] O. Bräysy, E. Martínez, Y. Nagata, and D. Soler. The mixed capacitated general routing problem with turn penalties. *Expert Systems with Applications*, 38(10):12954–12966, 2011.

[6] G. Capobianco, C. Cerrone, R. Cerulli, and G. Felici. Optimal Paths for Dual Propulsion Vehicles on Real Street Network Graphs. *International Conference on Optimization and Decision Science*, 517–527, 2017.

[7] C. Cerrone, R. Cerulli, and B. Golden. Carousel greedy: a generalized greedy algorithm with applications in optimization. *Computers & Operations Research*, 85:97–112, 2017.

[8] N. Christofides, V. Campos, A. Corberán, and E. Mota. An algorithm for the rural postman problem on a directed graph. In *Mathematical Programming Study 26 (Netflow at Pisa)*, pages 155–166. Springer, 1986.

[9] J. Clossey, G. Laporte, and P. Soriano. Solving arc routing problems with turn penalties. *Journal of the Operational Research Society*, 52(4):433–439, 2001.

[10] Á. Corberán and G. Laporte. *Arc routing: problems, methods, and applications*. SIAM, 2013.

[11] Á. Corberán, R. Martí, E. Martínez, and D. Soler. The rural postman problem on mixed graphs with turn penalties. *Computers & Operations Research*, 29(7):887–903, 2002.

[12] M. Drexl. On the generalized directed rural postman problem. *Journal of the Operational Research Society*, 65(8):1143–1154, 2014.

[13] J. Edmonds and E. L. Johnson. Matching, Euler tours and the Chinese postman. *Mathematical Programming*, 5(1):88–124, 1973.

[14] S. Irnich. Solution of real-world postman problems. *European Journal of Operational Research*, 190(1):52–67, 2008.

[15] H. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[16] L. Levy. Private communication. *RouteSmart Technologies*, 2013.

[17] L. Meng, Z. Hu, C. Huang, W. Zhang, and T. Jia. Optimized route selection method based on the turns of road intersections: A case study on oversized cargo transportation. *ISPRS International Journal of Geo-Information*, 4(4):2428–2445, 2015.

[18] J. Micó and D. Soler. The capacitated general windy routing problem with turn penalties. *Operations Research Letters*, 39(4):265–271, 2011.

[19] R. Ruiz and T. Stützle. An iterated greedy heuristic for the sequence dependent setup times flowshop problem with makespan and weighted tardiness objectives. *European Journal of Operational Research*, 187(3):1143–1159, 2008.

[20] D. Soler, E. Martínez, and J. Micó. A transformation for the mixed general routing problem with turn penalties. *Journal of the Operational Research Society*, 59(4):540–547, 2007.

[21] S. Vanhove and V. Fack. Route planning with turn restrictions: A computational experiment. *Operations Research Letters*, 40(5):342–348, 2012.

[22] D. Villeneuve and G. Desaulniers. The shortest path problem with forbidden paths. *European Journal of Operational Research*, 165(1):97–107, 2005.