The Windy Prize-Collecting Rural Postman Problem: 
An Ant-Colony Based Heuristic

Satu Elisa Schaeffer · Roger Z. Ríos-Mercado · Elena Fernández

Received: date / Accepted: date

Abstract The Prize-collecting Rural Postman Problem, also known as the Privatized Rural Postman Problem, is an arc-routing problem where each demand edge is associated with a profit, which is collected once if the edge is served, independently of the number of traversals. Included edges incur in routing costs proportionally to the number of traversals. In this paper, we introduce the Windy Privatized Rural Postman Problem in which routing costs also depend on the direction of the traversals. For this problem, we propose a solution heuristic based on ant-colony optimization. The proposed method is capable of constructing profitable closed walks with low computational load. The quality of the obtained solutions can be assessed comparing their values to lower bounds.

Keywords Edge routing · Price collecting · Windy · Ant-colony optimization

Mathematics Subject Classification (2000) MSC 90C59 · MSC 90C35 · MSC 90C27
1 Introduction

In Arc Routing Problems (ARPs) customer demand is represented by a subset of edges of a given graph and it is usually assumed that all the customer demand has to be served. In Prize-Collecting Arc Routing Problems (PARPs), however, demand edges are not necessarily served: there is a profit associated with each demand edge and the profit from each served edge is collected once, independently of the number of times it is traversed. The Prize-Collecting Rural Postman Problem (PRPP) was introduced under the name of Privatized Rural Postman Problem in [Aráoz et al. (2006)] and further studied by [Aráoz et al. (2009a)] who proposed an algorithm for solving it.

As shown by [Aráoz et al. (2006)], the PRPP is an extension of the well-known Rural Postman Problem, which is simply obtained by setting the profits of all demand edges sufficiently large. A further PARP, which has recently been studied by [Franquesa (2008)] and [Aráoz et al. (2009a)], [Aráoz et al. (2013)], is the Clustered Prize-Collecting Arc Routing Problem (CPARP). In the CPARP, the connected components defined by demand edges are considered: if a demand edge is served, then all the demand edges of its component are served. Other types of ARPs with profits have been studied, for instance, by [Feillet et al. (2005)], by [Archetti et al. (2010)], and more recently, by [Arbib et al. (2014)] who incorporate location decisions in the model. For a more comprehensive review, the reader is referred to the work by [Archetti and Speranza (2014)].

Many ARPs have been studied on windy graphs. Windy graphs are undirected graphs having two non-negative values associated with each edge, representing the costs of traversing the edge in either direction. Windy ARPs constitute an important class of problems, as the windy version of an ARP is a generalization of its undirected, directed and mixed versions. A global overview of the Windy General Routing Problem which contains most of the studied windy ARPs with a single vehicle as particular cases is given by [Corberán et al. (2008)]. Recently, [Benavent et al. (2014)] addressed a multi-vehicle WRPP. We only know, however, of three works — by [Franquesa (2008)], by [Corberán et al. (2011)], and by [Aráoz et al. (2013)] — considering windy PARPs. In all of these three cases the studied problem is the CPARP.

In this work we introduce the Windy Prize-Collecting Rural Postman Problem (WPRPP), which is the asymmetric version of the PRPP, and present a heuristic for it. To the best of our knowledge this is the first work on the WPRPP. The motivation for our study comes from the difficulty of the WPRPP as well as from its potential applications. As mentioned by [Aráoz et al. (2009a)], typically PARPs appear in the context of private companies looking to maximize operational profits, so that demand edges will not be served unless they yield a profit for the company, and each demand edge is served at most once. Applications of the WPRPP arise in the case of garbage collection, collection of goods for recycling or street cleaning, among others.

We approach the problem with a heuristic inspired by ant-colony optimization ([Dorigo and Birattari (2010)]) and report experiments regarding the tuning of the few parameters (mostly related to pheromone management). The exper-
imental results are satisfactory in terms of both solution quality and runtime, especially taking into account the difficulty of the problem.

The remainder of the paper is organized as follows. In Section 2 we formally introduce the WPRPP by means of a combinatorial description and develop some upper and lower bounds for the problem that will be useful for solution quality evaluation. In Section 3 we present the proposed heuristic. Section 4 describes the computational experiments that we have run, and gives extensive numerical results, which are thoroughly interpreted. The paper ends in Section 5 with some comments and conclusions.

2 Problem Description

In this section we introduce the WPRPP and give some bounds for it. First we provide some basic graph-theoretical definitions that are needed for the discussion. A graph is a pair of sets \((V, E)\), where the elements of \(V\) are called vertices and the elements of \(E\), which are called edges, are pairs of vertices. The number of vertices is denoted by \(n\) and the number of edges by \(m\); the density of a graph is the proportion of edges present from the maximum possible. The set of vertices that are connected to vertex \(v\) by edges that begin at \(v\) is called the neighborhood of \(v\) and denoted by \(\Gamma(v)\).

An edge can be directed meaning that the direction of traversal over that edge is fixed. The edge between vertices \(v\) and \(w\) is written as \(\{v, w\}\) when the traversal direction is indifferent, but as \((v, w)\) when it is fixed to originate from \(v\) and to end in \(w\), in which case it is called an arc.

A walk is a sequence of vertex visits that proceeds along edges of the graph, proceeding from a vertex to a neighboring vertex, and so forth. It is closed if it ends in the same vertex where it began; if there are no repeated vertices on a closed walk, it is a tour. A walk may be represented either as an ordered list of vertices that represents the visits or as an ordered list of arcs that represents the traversals.

2.1 Problem definition

**Input:** An undirected graph \(G = (V, E)\), with \(|V| = n\) and \(|E| = m\), a non-negative symmetric profit function \(p\) that assigns to each edge in \(E\) a value in \(\mathbb{R}\), a non-negative asymmetric cost function \(c\) that assigns to each edge in \(E\) two values in \(\mathbb{R}\) (one for each possible direction of traversal), and a depot vertex \(d \in V\).

**Task:** Find a closed walk \(T\) in \(G\) passing through \(d\) such that the total profit minus the cost is maximized. The cost is incurred at each traversal of an edge whereas the profit is collected only at first traversal of an edge. That is, the objective of the WPRPP is to maximize

\[
\max_{T \in \Pi_G} \{P(T) - C(T)\},
\]
where \( \Pi^G \) is the set of all possible closed walks in graph \( G \), \( P(\mathcal{T}) \) is the total profit gained by \( \mathcal{T} \) and \( C(\mathcal{T}) \) is the total cost incurred by \( \mathcal{T} \). Clearly, feasible walks \( \mathcal{T} \in \Pi^G \) with positive values of the objective \( P(\mathcal{T}) - C(\mathcal{T}) \) indicate profitable solutions.

Let \( t_{vw}^T \) denote the number of times that an arc \( (v, w) \) is traversed on closed walk \( \mathcal{T} \). Note that \( t_{vw}^T = 0 \) when \( \mathcal{T} \) does not contain the arc \( (v, w) \). Clearly, \( t_{vw}^T \) depends on \( \mathcal{T} \). When the closed walk \( \mathcal{T} \) we are referring to is clear from the context, we drop the index \( \mathcal{T} \).

Then, the total profit \( P(\mathcal{T}) \) is given by
\[
  P(\mathcal{T}) = \sum_{\{v, w\} \in E} \min \{1, t_{vw} + t_{wv}\} p_{vw}. \tag{2}
\]
Similarly,
\[
  C(\mathcal{T}) = \sum_{\{v, w\} \in E} t_{vw} c_{vw} + t_{wv} c_{wv} \tag{3}
\]
represents the total traversal costs. The WPRPP is NP-hard since it is a directed version of the PRPP, which is known to be NP-hard \cite{araoz2006}.\[2.2\] Bounds for WPRPP

We define an asymmetric auxiliary traversal-benefit function
\[
  \beta_{vw} = p_{vw} - c_{vw}, \tag{4}
\]
representing the net benefit of a first traversal of each arc, profitable if positive. Note that each edge in the graph gives rise to two arcs, for both of which the traversal-benefit is defined.

We also define a symmetric auxiliary benefit function \( \alpha \)
\[
  \alpha_{vw} = p_{vw} - (c_{vw} + c_{wv}) \tag{5}
\]
for each edge of the graph.

To derive an upper bound for the net total cost, we first associate with each edge \( \{v, w\} \in E \) the maximum possible net profit associated with it, namely \( \max\{\beta_{\{v, w\}}, \beta_{\{w, v\}}\} \). Then, an upper bound on the optimal WPRPP value can be obtained by considering only those edges with a positive net profit:
\[
  \mathcal{U} = \sum_{\{v, w\} \in E} \max\{0, \beta_{vw}, \beta_{wv}\}, \tag{6}
\]

To derive a lower bound, we obtain a solution in which the entire graph is traversed by depth-first search — by definition, each edge is traversed exactly once in each direction. The total profit of such walk gives the lower bound:
\[
  \mathcal{L} = \sum_{\{v, w\} \in E} \alpha_{vw}. \tag{7}
\]
In practice, we have observed that $\mathcal{L}$ is a very weak lower bound, and thus we have used a reference value $\mathcal{R}$ given by

$$\mathcal{R} = \sum_{\{v,w\} \in E} \max\{0, \alpha_{vw}\}$$

(8)

to contrast the quality of the heuristic solutions. Note that $\mathcal{R}$ accounts for the overall net profit that could be obtained with the profitable two-way traversal edges. Note that this is not a valid bound, neither lower nor upper.

Recall that $t$ is a function that maps each directed edge in $G$ to the number of times it is included in $T$. Now, the total benefit of a closed walk is

$$B = \sum_{(v,w) \in T} (p_{vw} - t_{vw}c_{vw}),$$

(9)

that is, all the involved costs minus the obtained profits. In order to obtain indicators of the quality of solutions that are comparable over all possible input graphs, we normalize both the reference value $\mathcal{R}$ of Equation (8) and the total benefit of a solution $B$ of Equation (9) in terms of the upper and lower bounds: this is done simply by mapping them in the $[0,1]$ interval as follows:

$$\hat{\mathcal{R}} = \frac{\mathcal{U} - \mathcal{R}}{\mathcal{U} - \mathcal{L}}$$

(10)

and

$$\hat{B} = \frac{\mathcal{U} - B}{\mathcal{U} - \mathcal{L}}.$$ 

(11)

For instance, if $\mathcal{L} = 50$ and $\mathcal{U} = 100$, a reference value $\mathcal{R} = 70$ would normalize to $\hat{\mathcal{R}} = 0.6$ and a solution with $B = 85$ would normalize to $\hat{B} = 0.3$. Note that the lower the value of $\hat{B}$ the better as it reflects how close the value of the solution is from its upper bound. If $\hat{B} > 1$ then the solution is very poor as even a full traversal is more cost effective. No good quality heuristic should produce such results. Using these normalized indices permits comparison between different instances of different size.

3 Ant-Colony Based Heuristic

Ant-colony optimization (ACO) (Dorigo and Birattari, 2010; Dorigo and Blum, 2005) refers these days to a very diverse set of agent-based heuristic algorithms that solve complex problems by performing — either sequentially or in parallel — several local searches that share information with simultaneous or future search agents through a (globally accessible) data structure referred to as a pheromone table (following the terminology common for ACO literature). Pheromone is deposited to mark promising regions of the solution space when visited by a single agent with the goal of attracting other agents towards it. This pheromone then evaporates over time. This type of methods have been successful in routing problems and are often employed (Ding et al., 2012; Narasimha et al., 2013; Reed et al., 2014; Ting and Chen, 2013).
Algorithm 1 A pseudocode of the selection of the next vertex to visit that takes as parameters $\gamma$ and $\epsilon$. The graph $G$ is given as input, as well as the current vertex $v$ and the present walk $T$ of length $\ell$ starting at $d$.

1: $\omega \leftarrow 0$ (total cost of the current walk)
2: for $w \in \Gamma(v)$ do
3: $\mathcal{E} \leftarrow \# \text{ of traversals of } \{v, w\}$ along $T$ in either direction
4: $\mathcal{V} \leftarrow \# \text{ of visits to } w \text{ along } T$
5: $\zeta_{vw} \leftarrow c_{vw}$
6: if $\mathcal{E} = 0$ then
7: $\zeta_{vw} \leftarrow \zeta_{vw} - p_{vw}$
8: end if
9: $\tau_{vw} \leftarrow \text{current pheromone level for } (v, w)$
10: $\nu_{vw} \leftarrow \tau \times \gamma + \zeta_{vw} \times (1 - \gamma)$
11: if $\nu_{vw} < 0$ then
12: $\nu_{vw} = \epsilon$
13: end if
14: $\chi_{vw} \leftarrow \tau_{vw}/(\mathcal{E} \times \mathcal{V} + 1)$
15: $\omega \leftarrow \omega + \tau_{vw}$
16: store $\tau_{vw}$ for arc $(v, w)$
17: end for
18: $\epsilon \leftarrow \text{Uniform}[0, \omega]; \alpha \leftarrow 0$ (roulette-wheel selection)
19: for $w \in \Gamma(v)$ in random order do
20: $\alpha \leftarrow \alpha + \tau_{vw}$
21: if $\alpha \ge \epsilon \lor (w = d \land \exp(\ell^{-1}) > \text{Uniform}[0, 1])$ then
22: return $w$
23: end if
24: end for

At the initialization step of the heuristic, we set a pheromone table at all-zero initial values: $\tau_{vw} = 0$ for all $(v, w)$. The same table will be used for all iterations (that is, for each ant). We begin each iteration of the heuristic with a walk consisting only of the depot vertex, $v = d$.

Tour extension. A new vertex to visit along the walk is chosen as follows: each neighbor $w$ of the current vertex $v$ is a candidate and it is given a preference weight that depends on the following factors:

- the number of times the arc $(v, w)$ has been included thus far, $t_{vw}$,
- the number of times the arc $(w, v)$ has been included thus far, $t_{wv}$,
- number of times the vertex $w$ has been visited thus far, denoted here by $\eta_w$,
- the traversal cost of the arc $(v, w)$, $c_{vw}$, and the
- the current level of pheromone for arc $(v, w)$, $\tau_{vw}$.

We compute

$$\nu_{vw} = -(1 - \gamma) \times \zeta_{vw} + \gamma \times \tau_{vw},$$

(12)

where $\gamma \in (0, 1)$ is a parameter controlling the importance of the pheromone table in this phase and

$$\zeta_{vw} = \begin{cases} c_{vw} - p_{vw}, & \text{if } t_{vw} + t_{wv} = 0, \\ c_{vw}, & \text{otherwise.} \end{cases}$$

(13)
We then compute the final preference weight as
\[ \chi_{vw} = \frac{\nu_{vw}}{(t_{vw} + t_{wv})\eta_{w} + 1}, \]
with
\[ \nu_{vw} = \begin{cases} \nu_{vw}, & \text{if } \nu_{vw} > 0, \\ \epsilon, & \text{otherwise,} \end{cases} \]
where \( \epsilon > 0 \) is a default-preference parameter in order to maintain all neighbors as potential next vertices and not create dead ends for the construction.

The goal of Equation (14) to make frequently visited vertices and multiply traversed edges less preferable (the constant one is added in the denominator to avoid division by zero when the neighbor has not yet been visited). We then perform a roulette-wheel selection using the values \( \chi_{vw} \) of the neighbors to select the neighbor to which to proceed along the walk. If the depot belongs to the candidate list, its preference is modified by increasing its selection probability by
\[ \exp(-\ell^{-1}), \]
where \( \ell \) is the length of the current walk, making it more probable to close the walk the longer it gets. The selected vertex is then appended to the traversal and the walk becomes one edge longer. The pseudocode of the selection process is described in Algorithm 1.

The main procedure of the ACO adaptation for the problem is shown in Algorithm 2 that incorporates the pheromone management. We keep the total cost of the current walk updated at all times (cf. lines 5 and 7 of Algorithm 2), as described below. This algorithm defines the following stopping criteria. If the last added vertex is the depot, we conclude the iteration as a closed walk has been completed. Otherwise, if the walk length already exceeds \( 2m \), we perform a cutoff to abort the iteration (lines 24–25 in Algorithm 2). This is to avoid getting stuck adding long loops; in our experiments we document the frequency with which this happens.

As we bind from above the maximum length of a potential solution and each step only examines the neighbors of the current vertex, a single iteration takes \( \mathcal{O}(m) \) time. In practice, the probability of not returning to the depot within the allowed length, is small, as will be discussed in Section 4. We also discuss the empirical runtime in detail in that section.

**Pheromone management.** When the selected vertex was already visited in the current partial solution, and the newly generated circuit is profitable, meaning that the total cost at the present \( B \) is negative, all the included edges increase their pheromone by \( \log(1 - B)/\ell \), where \( \ell \) is the length of the walk that constitutes the current partial solution and the constant one is present to ensure that it will always be a non-negative variation in the pheromone. Before carrying out a pheromone increase, the existing pheromone values are evaporated multiplying by a parameter \( \rho \in (0, 1) \).
Algorithm 2 A pseudocode of the proposed method, taking as parameters ρ, k, and ξ. The graph G together with a depot vertex d are given as input and the selected closed walk is produced as output. We abuse the notation for brevity treating the walk as a sequence and a set simultaneously.

1: pheromone table ← empty; a ← 0; ℓ ← 0;
2: B∗ ← nil; T∗ ← nil; (best seen)
3: while a ≤ k do
4: V = {d}; B ← 0; v ← d; T ← [d]
5: w ← select a neighbor of v (using Algorithm 1)
6: B ← B − c_{vw}
7: if (v, w) /∈ T ∧ (w, v) /∈ T then
8: B ← B + p_{vw}
9: T ← T appended by (v, w);
10: ℓ ← ℓ + 1
11: end if
12: if v ∈ V then
13: if B ≥ 0 then (profitable walk)
14: multiply the pheromone table by ρ to evaporate
15: end if
16: end if
17: V ← V ∪ {v}
18: if v = d then
19: if B > B∗ then
20: if B − B∗ ≥ δ then
21: a ← 0
22: else
23: a ← a + 1
24: end if
25: B∗ ← B
26: T∗ ← T
27: end if
28: else if ℓ > 2m then
29: restart at line 1 (a cutoff)
30: end if
31: end while
32: return T∗

A pseudocode of the main algorithm, incorporating the pheromone management to the walk extension and the iterations, is given in Algorithm 2. The stopping condition is discussed in the next section; for performing a fixed number of iterations, simply do not reset the counter a.

4 Experiments

All experiments are executed on a MacBook Air with a 1.4 GHz Intel Core 2 Duo processor, 4 GB of 1,067 MHz DDR3 memory, and a SSD, running OS 10.9.1, using the system-provided version of Python 2.7.2, using NumPy (http://www.numpy.org/) in addition to standard libraries. We have used the following set of instances, all of which are in turn generated from well-known Rural Postman Problem (RPP) or General Routing Problem (GRP) instances:
S1: The 118 CPARP instances used by Aráoz et al. (2009a) with symmetric costs.
S2: The 118 CPARP instances used by Aráoz et al. (2009a) modified to have asymmetric costs.
S3: The 40 GRP instances of http://www.uv.es/corberan/instancias.htm transformed into CPARP instances with symmetric costs.
S4: The 40 GRP instances of http://www.uv.es/corberan/instancias.htm transformed into CPARP instances with asymmetric costs.

Table 1 depicts information on the instances, grouped according to their characteristics and sizes. In each row, the instance group name, the number of instances (#) per group, number of nodes, and number of edges in the original graph, are displayed. The sets are described in more detail in Corberán et al. (2011).

<table>
<thead>
<tr>
<th>Group name</th>
<th>#</th>
<th>V</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Albaidaa</td>
<td>1</td>
<td>102</td>
<td>160</td>
</tr>
<tr>
<td>Albaidab</td>
<td>1</td>
<td>90</td>
<td>144</td>
</tr>
<tr>
<td>P</td>
<td>24</td>
<td>7–50</td>
<td>10–184</td>
</tr>
<tr>
<td>D16</td>
<td>9</td>
<td>16</td>
<td>31–32</td>
</tr>
<tr>
<td>D36</td>
<td>9</td>
<td>36</td>
<td>72</td>
</tr>
<tr>
<td>D64</td>
<td>9</td>
<td>64</td>
<td>128</td>
</tr>
<tr>
<td>D100</td>
<td>9</td>
<td>100</td>
<td>200</td>
</tr>
<tr>
<td>G16</td>
<td>9</td>
<td>16</td>
<td>24</td>
</tr>
<tr>
<td>G36</td>
<td>9</td>
<td>36</td>
<td>60</td>
</tr>
<tr>
<td>G64</td>
<td>9</td>
<td>64</td>
<td>112</td>
</tr>
<tr>
<td>G100</td>
<td>9</td>
<td>100</td>
<td>180</td>
</tr>
<tr>
<td>R20</td>
<td>5</td>
<td>20</td>
<td>37–75</td>
</tr>
<tr>
<td>R30</td>
<td>5</td>
<td>30</td>
<td>70–112</td>
</tr>
<tr>
<td>R40</td>
<td>5</td>
<td>40</td>
<td>82–203</td>
</tr>
<tr>
<td>R50</td>
<td>5</td>
<td>50</td>
<td>130–203</td>
</tr>
<tr>
<td>Alba, 3</td>
<td>5</td>
<td>116</td>
<td>174</td>
</tr>
<tr>
<td>Alba, 5</td>
<td>5</td>
<td>116</td>
<td>174</td>
</tr>
<tr>
<td>Alba, 7</td>
<td>5</td>
<td>116</td>
<td>174</td>
</tr>
<tr>
<td>GRP</td>
<td>10</td>
<td>116</td>
<td>174</td>
</tr>
<tr>
<td>Madr, 3</td>
<td>5</td>
<td>196</td>
<td>316</td>
</tr>
<tr>
<td>Madr, 5</td>
<td>5</td>
<td>196</td>
<td>316</td>
</tr>
<tr>
<td>Madr, 7</td>
<td>5</td>
<td>196</td>
<td>316</td>
</tr>
</tbody>
</table>

These sets of instances were further grouped into two subsets A and B according to their size. The 36-instance subset A has mostly smaller graphs, with \( n \in [7, 20] \), average being 15, and \( m \in [10, 40] \), the average being 27.
The 280-instance subset $B$ has mostly larger graphs, with $n \in [7, 196]$, average being 71, with $m \in [10, 316]$, the average being 134. The smaller set $A$ is here used for more extensive experimentation such as parameter-space exploration.

Each graph instance provides us with several input instances to our problem, as we use an input pair $(G, d)$, selecting a specific vertex to be the depot. Hence a single $n$-vertex graph in fact provides us with $n$ inputs for the experiments. In all our reported experiments, each vertex was used as depot.

Table 2 shows the relation between our sets $A$ and $B$ and the groups used in previous literature (Aráoz et al., 2013); our total also includes additional variants of sets $S_1$ through $S_4$ of the table, for which $|A| + |B| > \sum_i |S_i|$.

Three simple rules were used to generate the edge profits in the cases where none were provided in the original instance. All three rules generate the profit for each edge uniformly at random in an interval

$$p_{vw} \in [u, 3u),$$

where the difference is the value used for $u$. For the first profit-assignment scheme, we used the minimum of the two traversal costs defined for the edge in the original problem instance:

$$u = \min\{c_{vw}, c_{wv}\},$$

making it possible that some of the edges present are not profitable even when traversed in a single direction. In the second scheme, we use average:

$$u = \frac{1}{2}(c_{vw} + c_{wv}),$$

where it is still possible for an edge to be profitable in only one direction, even though less likely if the two costs differ (i.e., the windy characteristic we wish to attend). The third rule uses the maximum cost,

$$u = \max\{c_{vw}, c_{wv}\},$$

introducing possibly more edges with a higher traversal profit. It is no longer possible in this third scenario to have an edge that yields no profit with a single traversal in either direction.

Table 2: The first column indicates the instance group; $S_1$ and $S_3$ are instances with symmetric costs whereas $S_2$ and $S_4$ have asymmetric costs. $S_1$ and $S_2$ are sets ALBAIDA, P, D, G, and R, whereas $S_3$ and $S_4$ are ALBA, GRP, and MADR (Corberán et al., 2011).

<table>
<thead>
<tr>
<th>Group</th>
<th>$A$</th>
<th>$B$</th>
<th>Avg. $n$</th>
<th>Avg. $m$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_1$</td>
<td>18</td>
<td>100</td>
<td>45.6</td>
<td>96.7</td>
</tr>
<tr>
<td>$S_2$</td>
<td>18</td>
<td>100</td>
<td>45.6</td>
<td>96.7</td>
</tr>
<tr>
<td>$S_3$</td>
<td>0</td>
<td>40</td>
<td>146.0</td>
<td>227.3</td>
</tr>
<tr>
<td>$S_4$</td>
<td>0</td>
<td>40</td>
<td>146.0</td>
<td>227.3</td>
</tr>
</tbody>
</table>
4.1 Heuristic Fine-Tuning

We study the effects of the parameter values. First we explore values for the pheromone-weight factor

\[ \gamma \in \{0.0, 0.1, 0.2, \ldots, 1.0\}. \]  

The other parameters of the algorithm were fixed as follows: the inclusion constant \( \epsilon = 1 \) and the evaporation factor \( \rho = 0.9999 \). We perform 30 executions on the \( A \) set, using now only the minimum profit-assignment scheme of Equation (18). We measured the distribution of the normalized best total cost \( \hat{B} \) of Equation (11) over the set of executions of the heuristic for each value of \( \gamma \). The results are shown in Figure 1a.

As can be seen, the minimum of the best solution value remains very small on all parameter values, but the behavior of the worse solutions varies. In order to select the best value for the parameter, we observe that the medians of \( \hat{B} \) are smaller than 0.3 for \( \gamma < 0.5 \) and slowly rise above that for larger values. Also the minima of \( \hat{B} \) are less than 0.01 for \( \gamma < 0.4 \) and then grow. The value 0.2 minimizes the first quartile and is the second smallest for the third; thus we choose to fix \( \gamma = 0.2 \) for the remaining experiments.

Now, we proceed to vary the evaporation factor of the pheromone table

\[ \rho \in \{0.1, 0.2, 0.4, 0.6, 0.8, 0.9, 0.99, 0.999, 0.9999\}. \]  

We include several values close to one as the evaporation in many other ACO-type heuristics is kept slow. A value of 0.7 is recommended by [Colorni et al. 1992], but we are particularly keen to direct the future walks towards observed profitable regions as we aim to combine as many as possible, for which we chose a very high value in the first place. The rest of the experimental setup remains unaltered; the results are shown in Figure 1b, where we report the normalized
best total cost $\hat{B}$ of Equation (11) over the set of executions of the heuristic for each value of $\rho$.

The best median results are obtained with $\rho \geq 0.8$, supporting the initial hypothesis of slow evaporation being the most favorable. For keeping the worst results at the best possible level as well as making the best results the best possible ones, we choose $\rho = 0.9999$, as originally. Again, the runtimes of individual executions for input pairs $(G,d)$ remained at fractions of a second for all values of the parameter.

Finally, fixing all other parameters to the values chosen in this section, we study the effect of the inclusion constant $\epsilon \in \{2^{-6}, 2^{-5}, \ldots, 2^{8}, 2^9\}$. (23)

The maximum value is set relative to the largest edge cost present in set $A$, which is 305, thus making the highest possible gain — cf. Equation (17) — for a first traversal $610 < 1,024 = 2^{10}$. Using values of a magnitude similar to the largest possible gain would do little to guide the heuristic and are therefore omitted.

The results are shown in Figure 2. For $\epsilon > 2^2$, the best results begin to degrade. The medians of $\hat{B}$ are at their best for $\epsilon < 2^{-2}$, but the number of cutoffs (that is, walks that reach length $2m$ before returning to the depot as indicated in Algorithm 2) begins to rapidly grow for $\epsilon < 2^{-3}$. Hence we use $\epsilon = 2^{-3} = 0.0125$ for the remaining experiments, keeping the number of cutoffs at two at the most for each of the 30 executions.

We now estimate the number of iterations after which we can normally expect not to further improve the current best solution. The results are shown in Figure 3 and reveal that the heuristic keeps on improving upon the best solution for quite a large number of iterations. However, the magnitude of the improvement drops drastically already after 20–30 iterations, as shown in Figure 4. However, there are sudden jumps later down in all three plots of Figure 4, and therefore, instead of fixing the iteration count per se, we use a flexible stopping criterion, as the improvement rate seems to depend on the instance structure: as soon as $k = 30$ consecutive iterations have not reached

Fig. 2: On the horizontal axis, the power of two $x$ used the inclusion constant $\epsilon = 2^x$, and on the vertical axis, box plots of the best normalized total cost $\hat{B}$. 
Fig. 3: A histogram showing the iteration number (horizontal axis) at which the best normalized total cost $\hat{B}$ was found, up to a maximum of 500 iterations; the vertical axis indicates the frequency.

an improvement over $\xi = 0.05$ on the normalized cost of the best closed walk, the heuristic stops; the pseudocode in Algorithm 2 reflects this setup.

In summary, we have found thus far with the experiments on set $A$ that for the minimum profit-assignment scheme, choosing the parameters as $\gamma = 0.2$, $\rho = 0.9999$ and $\epsilon = 0.125$ is a functional setup and that we may cease when 30 iterations have failed to produce at least a 0.05 improvement on the normalized total cost. For the remainder of the experiments we fix our heuristic parameters at these values.

4.2 Heuristic Assessment on Small Graphs

We ran the heuristic first for all of the input graphs in set $A$, using all three profit-generation methods. For each execution, we recorded the values of the upper and lower bounds $U$ and $L$, respectively, as well as the reference value $R$, together with the numbers of edges included in the sums of $L$ and $R$ — see Equations (7) and (8), respectively. We computed for each input pair $(G,d)$ the total cost of the best closed walk $B^*$, the length of this best closed walk found, and the time in seconds it took to compute it.

Figure 5 shows the quality of the results obtained over set $A$. It can be seen that the values of the obtained solutions are in most cases much closer to the lower bound than to the upper bound, improving on the reference value quite significantly in many cases. The gap between the reference value and that of the value of the obtained solution varies quite a lot from one instance to another, making it clear that the instance structure has a significant effect on the difficulty of the instance (also the reference values are similarly affected).

Changing the profit-assignment scheme alters the abundance of edges that are profitable regardless of the direction of traversal. The proposed heuristic performs better under the minimum-profit assignment scheme of Equation 18 (the left plot in Figure 5) than when the instance becomes “easy” in the sense that all edges are profitable for a single traversal, meaning that $p_{vw} \geq \max\{c_{vw}, c_{wv}\}$ — but not necessarily double traversal; cf. Equation 20. On the horizontal axis in Figure 5 for each instance, the results corresponding to its different possible depots are depicted consecutively.
(a) The number of times that an improvement was obtained on each iteration (on the vertical axis).

(b) The average (drawn as dots) and standard deviation (shown with vertical error bars) of the improvements obtained on each iteration (on the vertical axis), using a linear scale on the horizontal axis.

(c) The average (drawn as dots) and standard deviation (shown with vertical error bars) of the improvements obtained on each iteration (on the vertical axis), using a logarithmic scale on the horizontal axis.

Fig. 4: The improvement obtained on each iteration (horizontal axis) for all graphs in set $A$.

Figure 6 shows the lengths of the obtained solutions divided by $2m$ (i.e., the total length of a depth-first search), expressed as percentages — values above 100 are possible for closed walks that traverse at least some edges more than twice.

We visualize the execution time of the proposed heuristic on the $A$ set in Figure 7. The horizontal and vertical axes indicate the number of vertices ($n$)
Fig. 5: On the vertical axis, the normalized reference values $\hat{R}$ and the normalized values of the best obtained solutions — cf. Equations (10) and (11), respectively — for the three profit-assignment schemes in set $A$. The size of each dot is proportional the number of edges included in the computation divided by $2m$, that is, the larger it is, the bigger proportion of the edges has been included. The light squares correspond to the reference values and the dark circles to those of the obtained solutions.

and the number of edges ($m$), respectively. For the sake of a visual comparison computing times are proportional to dot diameters. Each of the three profit-assignment schemes is drawn separately.

For the numerical values of the runtimes, Figure 8 shows the individual runtime histograms for each of the three profit-assignment schemes, using the total time for all the iterations until reaching the stopping condition. In each case, for each instance and possible depot, its runtime indicates the overall computing time (in seconds) for all the iterations until the stopping condition was reached. It can be observed that the profit-assignment scheme has little effect on the runtime. The runtime over the 3,360 executions was over two seconds in only twelve cases and over five seconds only twice: 6.5 seconds once under the average scheme and 10.6 seconds once under the average scheme. For an improved visibility of the small runtimes we have cut the long tail from the histogram.

4.3 Heuristic Assessment on Larger Graphs

We now report the behavior of the proposed heuristic on instances of set $B$, using the same parameter values as above. For these graphs, we experimented only under the minimum profit-assignment scheme of Equation (18), and also counted the number of iterations aborted and the total time spent on the aborted iterations.

First, in Figure 9a, we show the quality of the solutions themselves by means of the normalized total cost of Equation (11). These are all well beneath
Fig. 6: Histogram of the normalized lengths of closed walks, using each vertex in turn as depot.

the upper bound and concentrated around the value $L + \frac{1}{2}(U - L)$. In the vast majority of cases the values of the obtained solutions are much better than the reference values; cf. Equation (8). Only in a few pathological cases the values of the obtained solutions are worse than the corresponding reference values. We should also note that the instance structure affects the quality of the obtain solutions, as was the case with the instances in set $A$ in Figure 5.

Fig. 7: The maximum runtime (represented by the diameter of the dot) of the heuristic within set $A$ with respect to the number of vertices ($n$) and edges ($m$).
As we are now using a dynamic stopping condition based on the improvement obtained, we include in Figure 9c a histogram of the frequency with which the number of iterations reaches a given number. It rarely reaches 100. The runtimes for set $B$ are shown in Figure 9d. There is a flat heavy tail on the runtimes (shown in Figure 10): approximately one percent of the executions take over two minutes. For set $B$, we find that 84 percent of the times the solution is produced in less than 30 seconds and 95 percent of the times, the solution is obtained in less than one minute.

As the proposed heuristic is non-deterministic, a detailed asymptotic analysis can get lengthy, but we want to provide experimental evidence for the informal reasoning of the time complexity. In Figure 11 we plot the average runtime for the values of $n$ and $m$ that appear within set $B$. In Figures 11a and 11b, each value of $n$ or $m$, respectively, is considered independently (we call this the raw data), where fit attempts have high error. Figures 11c and 11d give similar results when the values of $n$ and $m$ are respectively binned. For this we have used fixed width bins of sizes 20 for $n$ and 70 for $m$.

As can be seen in Figure 11, the empirical time requirements of the heuristic can be better appreciated with the binned data than with the raw data. We also exclude in each case one outlier when fitting curves to the binned data (indicated in the figures). The cubic curve obtained for the binned data adjusts quite accurately for both $n$ and $m$, whereas the quadratic curve adjusts adequately only for $m$, indicating experimental time complexity of order $O(n^3)$ and $O(m^2)$ for these instances. In general, $m \in O(n^2)$ for graphs with densities close to one. However, graphs in set $B$ have, in general, a much lower density of edges which on average is $\approx 0.15$. This explains why we have observed in our experiments $O(m^2) \sim O(n^3)$.

As already mentioned, the heuristic terminates when the walk is longer than $2m$. We counted the number of such cutoffs and report the histogram of the strictly positive values in Figure 12. In total 91.4 % of the executions had zero cutoffs and are excluded from the histogram.

Fig. 8: Histograms of the individual runtimes over all executions of the heuristic for the three profit-assignment schemes.
4.4 Robustness of the Solutions Found

We also experimented on the closeness of the computed solution with respect to the best solution found during a set of iterations. For set $A$ (the smaller graphs) we executed 100 iterations for each input pair $(G, d)$, whereas for set $B$ we only executed 10 iterations for each input pair. We recorded for each pair which was the best solution $B^*$ obtained for that pair and computed for any other solution value $B$ that was obtained for that same pairs tolerance of

![Diagram showing exceedance percentage vs. runtime](image)

Fig. 10: Percentage of instances that exceed a given runtime for instances in set $B$. 

---

Fig. 9: Histograms of normalized costs (absolute and relative), numbers of iterations, and runtimes for set $B$. 

(a) The best normalized cost $100 \times \hat{B}$. (b) The ratio $100 \times \hat{B}/\hat{R}$. 

(c) Iterations per execution. (d) Total runtimes in seconds.
how far that value was from $B^*$:

$$\rho = \frac{B^* - B}{B^*}. \quad (24)$$

Note that there is no limit as such to the values that $B$ can take — the route may not have been profitable or have a profit (or a cost) near zero. Therefore, $\rho \geq 0$, but is not bounded from above.

It is important to remember that the costs and the benefits are floating-point numbers and the benefits were generated pseudo-randomly at uniform

Fig. 12: Histogram of the number of cutoffs per execution, when at least one cutoff was present (that is, in 8.6% of the executions).
over a continuous range, for which equality is unlikely unless the same exact solution is returned twice (which again is unlikely for a randomized heuristic). Thus we cannot reasonably count the number of times the best solution appeared, but rather how far from the best value were the other solution values. We illustrate this in terms of tolerance levels: we compute for a given tolerance $\kappa > 0$ the percentage of solutions values that were no further than $\kappa$ from the best solution.

We used values for $\kappa$ from 0.001 onward in multiples of two; the resulting plot is shown in Figure 13. Note that for set $A$, the best solutions necessarily comprised 1% whereas for set $B$ it was 10%, for which the $B$ set plots begin at a higher level. It can be observed in Figure 13 that the profit-assignment scheme (minimum, average, or maximum of the two costs defined in the instance) has a minor effect on the results (the minimum-generation being the hardest to make profitable and hence also presenting a heavier tail of worse solutions) and the shape of the curve is generally the same for the 100-execution $A$ set and the 10-execution $B$ set. The rapid increase indicates that there are several solutions reaching approximately half of the best profit.

5 Conclusions

In this work, we have introduced the Windy Prize-Collecting Rural Postman Problem and proposed an efficient heuristic for solving it, motivated by ant-colony optimization. The heuristic performs essentially weighted random walks on the graph, using a pheromone table to store information on profitable partial walks from early iterations towards future computations. We document numerical results from numerous computational experiments, where in most cases good solutions (i.e., profitable closed walks) are obtained, almost always in less than a half a minute for problem instances adapted from previous literature with up to 196 vertices and 632 edges.
In future work, parallelized versions of the heuristic are of interest, as it is rather straight-forward to implement on a multicore or a distributed system as long as the pheromone table can be stored for read-write access in shared memory and improvements on the best known solution are “broadcasted” for implementing the relative stopping condition.

We will also consider a complete restart when a higher number of cutoffs are made, so as to diminish the heavy tail reported for runtimes. We suspect that in some instance structures, the pheromones placed early on may direct the future constructions to regions away from the depot and then cycle there, avoiding return to the depot. We also leave the introductions of re-departures from the depot to future work, as it would require a roll-back mechanism for when profitable extensions are not found and hence somewhat complicates the formulation of the heuristic.

Another concern, with synergies to the parallel implementation, is the computational scalability of the proposed heuristic towards massive graphs. We have implemented an instance generator based on a 3D-landscape (a randomized fractal-like process) and modelling with simple physics the cost (work against gravity and friction) for creating windy graph instances and hope to use real-world data from cities for generating realistic profit assignments. This would permit the creation of arbitrary-sized problem instances.

Acknowledgments

The research of the second author was funded by the Mexican Council for Science and Technology through grant SEP-CONACyT CB-2011-01–166397 and by UANL under its Scientific and Technological Research Support Program through grant PAICYT CE728-11. The third author was partially supported by the Spanish Ministry of Science and Education through grant MTM2012-36163-C06-05 and by ERDF funds.

References


