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

Satu Elisa Schaeffer
FIME, UANL, Av. Universidad s/n, Cd. Universitaria,
San Nicolás de los Garza, Mexico
E-mail: *elisa.schaeffer@uanl.edu.mx*

Roger Z. Ríos-Mercado
Graduate Program in Systems Engineering,
Universidad Autónoma de Nuevo León, Av. Universidad s/n, Cd. Universitaria,
San Nicolás de los Garza, Mexico

Elena Fernández
Department of Statistics and Operations Research,
Universitat Politècnica de Catalunya, Barcelona, Spain

April 14, 2014

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 the bounds.

Keywords: Edge routing; Price collecting; Windy; Ant-colony optimization.

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. [2009b] 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 route-construction problems 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 experimental 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 mathematical formulation 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 en *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 $\mathcal{T}$ in G that includes a given depot vertex d maximining the total profit minus the cost, where a *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 function is

$$\max_{\mathcal{T} \in \Pi^{(G,d)}} \big(P(\mathcal{T}) - C(\mathcal{T})\big), \tag{1}$$

where $\Pi^{(G,d)}$ is the set of all possible closed walks in graph G that contain d, $P(\mathcal{T})$ is the total profit gained by $\mathcal{T}$, and $C(\mathcal{T})$ is the total cost incurred by $\mathcal{T}$. Walks $\mathcal{T} \in \Pi^G$ with positive values of the objective $P(\mathcal{T}) - C(\mathcal{T})$ indicate profitable solutions.

Let $t_{vw}^{\mathcal{T}}$ denote the number of times that an arc (v, w) is traversed on closed walk $\mathcal{T}$. Note that $t_{vw}^{\mathcal{T}} = 0$ when $\mathcal{T}$ does not contain the arc (v, w). Clearly, $t_{vw}^{\mathcal{T}}$ 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 [Aráoz et al., 2006].

2.2 Bounds for WPRPP

We define an asymmetric auxiliary traversal-profit function

$$\beta_{vw} = p_{vw} - c_{vw}, \tag{4}$$

representing the net profit 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-profit is defined.

We also define a symmetric auxiliary profit function α

$$\alpha_{vw} = p_{vw} - (c_{vw} + c_{wv}) \tag{5}$$

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

$$\mathcal{R} = \sum_{\{v,w\}\in E} \max\{0,\, \alpha_{vw}\} \tag{8}$$

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 $\mathcal{T}$. Now, the *total profit* of a closed walk is

$$\mathcal{B} = \sum_{(v,w)\in\mathcal{T}} \left(p_{vw} - t_{vw}c_{vw}\right), \tag{9}$$

that is, all the obtained profits minus the involved costs. 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 profit of a solution $\mathcal{B}$ of Equation (9) in terms of the upper and lower bounds:

$$\hat{\mathcal{R}} = \frac{\mathcal{U} - \mathcal{R}}{\mathcal{U} - \mathcal{L}} \tag{10}$$

and

$$\hat{\mathcal{B}} = \frac{\mathcal{U} - \mathcal{B}}{\mathcal{U} - \mathcal{L}}, \tag{11}$$

where the resulting value is in the [0, 1] interval when the reference value or the total profit, respectively, is between the two bounds, and exceeds one whenever the value in question is worse than the lower bound (which is possible for a heuristic unless the solutions are pruned specifically

to exclude solutions worse than the known bound). 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 $\mathcal{B} = 85$ would normalize to $\hat{\mathcal{B}} = 0.3$. Note that the lower the value of $\hat{\mathcal{B}}$ the better as it reflects how close the value of the solution is from its upper bound. If $\hat{\mathcal{B}} > 1$ then the solution is very poor as even a full traversal is more cost effective. Using these normalized indices permits comparison between different instances of different size and with very different values for the upper and lower bounds.

3 Ant-Colony Based Heuristic

Ant-colony optimization (ACO) [Dorigo and Birattari, 2010, Dorigo and Blum, 2005] refers 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].

In particular, we chose ACO for the present problem as the input graph can be thought to consist of profitable and non-profitable zones; a minimal profitable zone is a single edge that has a lower traversal cost (at least in one direction) than the profit obtained from it, and then larger profitable zones are formed as induced subgraphs of vertices that connected to each other with such profitable edges. A solution to the problem needs to pick some of these protifable zones and to connect them into a closed tour that includes the depot. The pheromone analogy of ACO that is intended to guide a search procedure towards high-fitness subsolutions, even when it is required to pass through a non-promising zone to reach them, is a good fit to the problem at hand.

Also Santos et al. [2010] use ACO for a different arc routing problem; their approach starts with a constructive phase of generating initial solutions which are then improved by pheromone-guided exploration. Our solution has no initial population but rather uses the pheromone table so that subsequent constructions may identify profitable zones based on computations done by earlier constructions. Also, their problem has capacity constraints regarding the tour to be constructed, whereas our problem poses to restriction to the length of the closed walk as long as extending the route is able to provide an increase in the total benefit. Doerner et al. [2004] solve a capacitated routing problem with an ACO as well, with promising results.

At the initialization step of our proposed 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 η_w,
- the traversal cost of the arc (v, w), c_{vw}, and the

Algorithm 1 A pseudocode of the selection of the next vertex to visit that takes as parameters γ and ϵ. The graph G is given as input, as well as the current vertex v and the present walk $\mathcal{T}$ of length ℓ starting at d.

$\omega \leftarrow 0$ (total cost of the current walk)
for $w \in \Gamma(v)$ **do**
  $\mathcal{E} \leftarrow$ # of *traversals* of $\{v, w\}$ along $\mathcal{T}$ in *either direction*
  $\mathcal{V} \leftarrow$ # of *visits* to w along $\mathcal{T}$
  $\zeta_{vw} \leftarrow c_{vw}$
  if $\mathcal{E} = 0$ **then**
    $\zeta_{vw} \leftarrow \zeta_{vw} - p_{vw}$
  end if
  $\tau_{vw} \leftarrow$ current pheromone level for (v, w)
  $\nu_{vw} \leftarrow \tau_{vw} \times \gamma + \zeta_{vw} \times (1 - \gamma)$
  if $\nu_{vw} < 0$ **then**
    $\nu_{vw} = \epsilon$
  end if
  $\chi_{vw} \leftarrow \nu_{vw}/(\mathcal{E} \times \mathcal{V} + 1)$
  $c_{vw} \leftarrow$ cost of travelsal for (v, w)
  $\omega \leftarrow \omega + c_{vw}$
  store τ_{vw} for arc (v, w)
end for
$c \leftarrow \text{Uniform}[0, \omega]$; $\alpha \leftarrow 0$ (*roulette-wheel selection*)
for $w \in \Gamma(v)$ in *random order* **do**
  $\alpha \leftarrow \alpha + \tau_{vw}$
  if $\alpha \geq c \vee \big(w = d \wedge \exp(-\ell^{-1}) > \text{Uniform}[0, 1]\big)$ **then**
    return w
  end if
end for

- the current level of pheromone for arc (v, w), τ_{vw}.

We compute

$$\nu_{vw} = \gamma \times \tau_{vw} + (1 - \gamma) \times \zeta_{vw} + \tag{12}$$

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

$$\zeta_{vw} = \begin{cases} c_{vw} - p_{vw}, & \text{if } t_{vw} + t_{wv} = 0, \\ c_{vw}, & \text{otherwise.} \end{cases} \tag{13}$$

We then compute the final preference weight as

$$\chi_{vw} = \frac{\upsilon_{vw}}{(t_{vw} + t_{wv})\eta_w + 1}, \tag{14}$$

with

$$\upsilon_{vw} = \begin{cases} \nu_{vw}, & \text{if } \nu_{vw} > 0, \\ \epsilon, & \text{otherwise,} \end{cases} \tag{15}$$

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 χ_{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

$$\exp(-\ell^{-1}), \tag{16}$$

Algorithm 2 A pseudocode of the proposed method, taking as input the current pheromone table $\mathcal{P}$, formed by the present values of τ_{ij}, ρ, the graph G together with a depot vertex d are given as input and a closed walk is produced as output. We abuse the notation for brevity treating the walk as a sequence and a set simultaneously.

1: $\mathcal{V} = \{d\}$
2: $\mathcal{B} \leftarrow 0$
3: $v \leftarrow d$
4: $\mathcal{T} \leftarrow [d]$
5: $\ell \leftarrow 1$
6: **while** (conditionless loop) **do**
7: $w \leftarrow$ select a neighbor of v (using Algorithm 1)
8: $\mathcal{B} \leftarrow \mathcal{B} - c_{vw}$
9: **if** $(v, w) \notin \mathcal{T} \wedge (w, v) \notin \mathcal{T}$ **then**
10: $\mathcal{B} \leftarrow \mathcal{B} + p_{vw}$
11: **end if**
12: $\mathcal{T} \leftarrow \mathcal{T}$ appended by (v, w)
13: $\ell \leftarrow \ell + 1$
14: **if** $v \in \mathcal{V}$ **then**
15: **if** $\mathcal{B} \geq 0$ **then** (profitable walk)
16: $\forall \tau_{ij} \in \mathcal{P}, \tau_{ij} \leftarrow \rho \times \tau_{ij}$ (pheromone *evaporation)*
17: $\tau_{ij} \leftarrow \tau_{ij} + \log(1 + \mathcal{B})/\ell, \forall (i, j) \in \mathcal{T}$ (pheromone *deposit*)
18: **end if**
19: **end if**
20: $\mathcal{V} \leftarrow \mathcal{V} \cup \{v\}$
21: **if** $v = d$ **then return** $\mathcal{T}$ (closed walk completed)
22: **else if** $\ell > 2m$ **then**
23: restart at line 1 (a *cutoff* for failing to close the walk)
24: **end if**
25: **end while**

where ℓ 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.

Algorithm 2 shows the construction of a single closed tour, whereas the main procedure of the ACO adaptation for the problem is shown in Algorithm 3. We use the following stopping criteria: if the last added vertex is the depot, we probabilistically either conclude the iteration (as a closed walk has been completed) or start another segment to extend it. If the last vertex was not the depot and the walk length already exceeds $2m$, we perform a cutoff, meaning that we abort and discard the iteration (lines 24–25 in Algorithm 3). 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 $\mathcal{B}$ is positive, all the included edges increase their pheromone by $\log(1 - \mathcal{B})/\ell$, where ℓ 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 $\rho \in (0, 1)$.

A pseudocode of the main algorithm, incorporating the pheromone management to the walk

Algorithm 3 A pseudocode of the proposed method, taking as input the maximum stall count k, the improvement threshold ξ, as well as the input graph G together with a depot vertex d. A closed walk is produced as output. We abuse the notation for brevity treating the walk as a sequence and a set simultaneously.

```
P ← empty (global pheromone table)
a ← 0 (stall counter)
B* ← nil (best total profit seen)
T* ← nil (best solution seen
while a ≤ k do
    compute T for (G, d) based on P (using Algorithm 2)
    B ← the total profit of T
    if B > B* then
        if B − B* ≥ ξ then
            a ← 0 (improvement resets the stall counter)
        else
            a ← a + 1
        end if
        B* ← B
        T* ← T
    end if
end while
return T*
```

extension and the iterations, is given in Algorithm 3. The stopping condition is discussed in the next section; when performing a fixed number of iterations, we simply never 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].

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

Table 1: Summary of the instances.

Group name	#	$\|V\|$	$\|E\|$
ALBAIDAA	1	102	160
ALBAIDAB	1	90	144
P	24	7–50	10–184
D16	9	16	31–32
D36	9	36	72
D64	9	64	128
D100	9	100	200
G16	9	16	24
G36	9	36	60
G64	9	64	112
G100	9	100	180
R20	5	20	37-75
R30	5	30	70-112
R40	5	40	82-203
R50	5	50	130-203
ALBA_3	5	116	174
ALBA_5	5	116	174
ALBA_7	5	116	174
GRP	10	116	174
MADR_3	5	196	316
MADR_5	5	196	316
MADR_7	5	196	316

$n \in [7, 196]$, 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 $S1$ through $S4$ 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

$$p_{vw} \in [u, 3u), \tag{17}$$

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:

$$u = \min\{c_{vw}, c_{vw}\}, \tag{18}$$

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:

$$u = \tfrac{1}{2}(c_{vw} + c_{wv}), \tag{19}$$

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,

$$u = \max\{c_{vw}, c_{wv}\}, \tag{20}$$

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.

4.1 Heuristic Fine-Tuning

We first study the effects of the parameter values with the goal of determining the ideal value to use for each; we leave autoadaptive parameter setting for future work.

First we explore values for the pheromone-weight factor

$$\gamma \in \{0.0, 0.1, 0.2, \ldots, 1.0\}. \tag{21}$$

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 $\hat{\mathcal{B}}$ of Equation (11) over the set of executions of the heuristic for each value of γ. 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 $\hat{\mathcal{B}}$ are smaller than 0.3 for $\gamma < 0.5$ and slowly rise above that for larger values. Also the minima of $\hat{\mathcal{B}}$ 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.

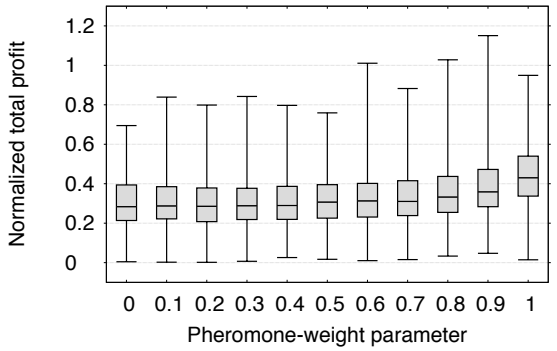
Now, we proceed to vary the evaporation factor of the pheromone table

$$\rho \in \{0.1, 0.2, 0.4, 0.6, 0.8, 0.9, 0.99, 0.999, 0.9999\}. \tag{22}$$

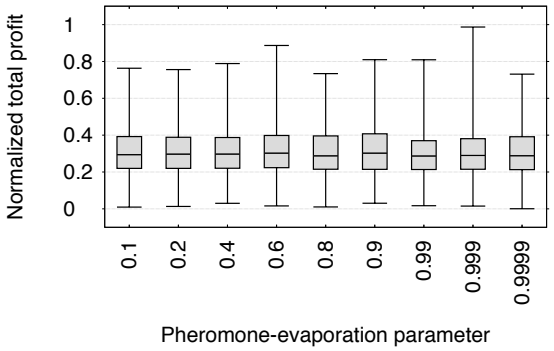
We include several values close to one as the evaporation in many other ACO-type heuristics is kept slow. A value of 0.7 is recommended by Colorni et al. [1992], but we are particularly keen

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].

Group	A	B	Avg. n	Avg. m
S_1	18	100	45.6	96.7
S_2	18	100	45.6	96.7
S_3	0	40	146.0	227.3
S_4	0	40	146.0	227.3



(a) On the horizontal axis, the pheromone-weight factor γ, and on the vertical axis, box plots of the best normalized total cost $\hat{\mathcal{B}}$.



(b) On the horizontal axis, the pheromone evaporation factor ρ, and on the vertical axis, box plots of the best normalized total cost $\hat{\mathcal{B}}$.

Figure 1: Effects of the pheromone-affecting parameters.

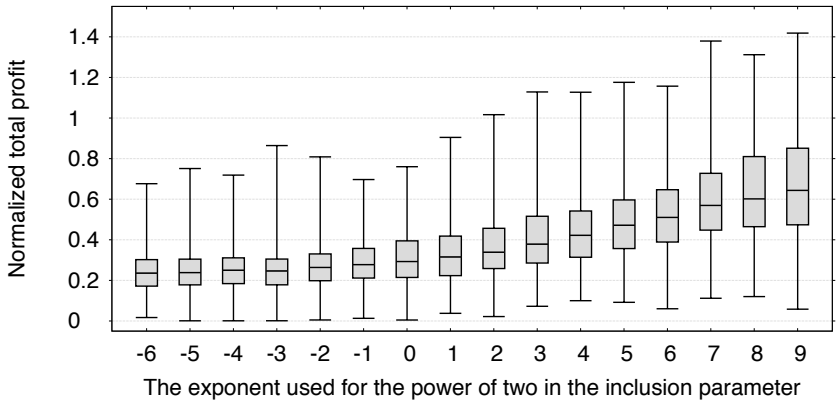


Figure 2: On the horizontal axis, the power of two x used the inclusion constant $\epsilon = 2^x$, and on the vertical axis, box plots of the best normalized total cost $\hat{\mathcal{B}}$.

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 $\hat{\mathcal{B}}$ of Equation (11) over the set of executions of the heuristic for each value of ρ.

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

$$\epsilon \in \{2^{-6}, 2^{-5}, \ldots, 2^8, 2^9\}. \tag{23}$$

The maximum value for ϵ was set to be just above the three times the largest edge cost present in set A, as the profits were generated within ranges depending on the costs (cf. Equation (17)); using values of a magnitude similar to the largest possible gain would do little to guide the heuristic and

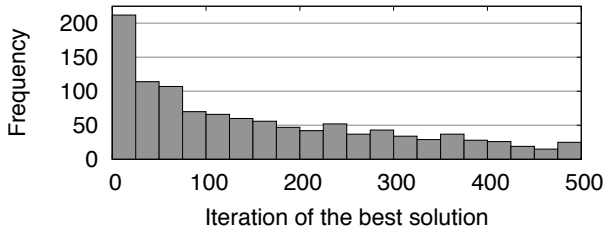


Figure 3: A histogram showing the iteration number (horizontal axis) at which the best normalized profit $\hat{\mathcal{B}}$ was found, up to a maximum of 500 iterations; the vertical axis indicates the frequency.

are therefore omitted.

The results are shown in Figure 2. For $\epsilon > 2^2$, the best results begin to degrade. The medians of $\hat{\mathcal{B}}$ 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 3) 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 an improvement over $\xi = 0.05$ on the normalized cost of the best closed walk, the heuristic stops; the pseudocode in Algorithm 3 reflects this setup.

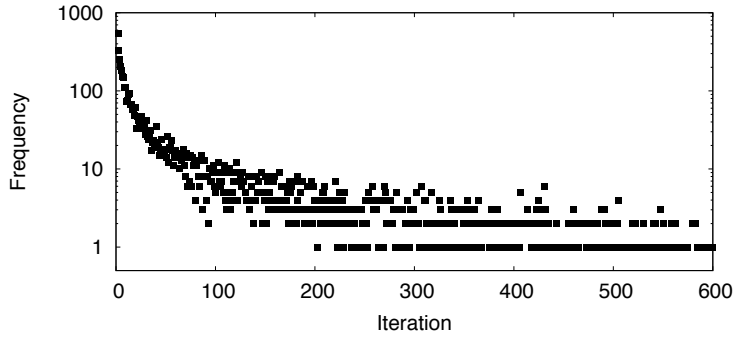
In summary, we have found thus far with the experiments on set A that for the minimum profit-assignment scheme, choosing the parameters as $\gamma = 0.2$, $\rho = 0.9999$ and $\epsilon = 0.125$ is a functional setup and that we may cease when 30 iterations have failed to produce at least a 0.05 improvement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

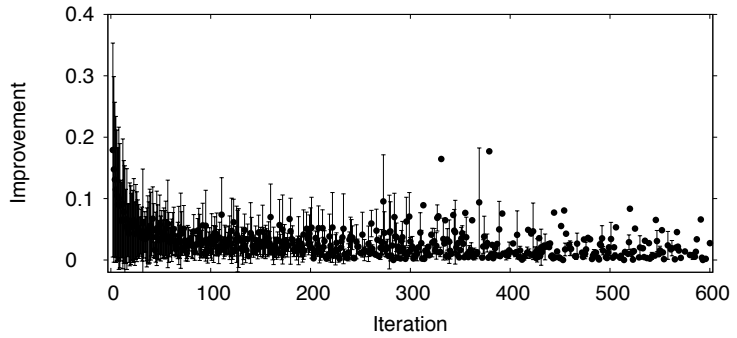
We ran the heuristic with the parameter values determined in the previous section first for all of the input graphs in set A, using all three profit-generation methods. Each vertex acted as depot once and ten replicas.

We also ran modified versions of the heuristic for comparison: a *simple* search that selects the next vertex to visit uniformly at random, a *greedy* search that select the next vertex to visit in such a way that more a candidate increases the total profit of the present tour fragment, the higher the probability of selecting it, and a *tabu* search with recently used edges being blocked from being used again, unless no other candidate is present.

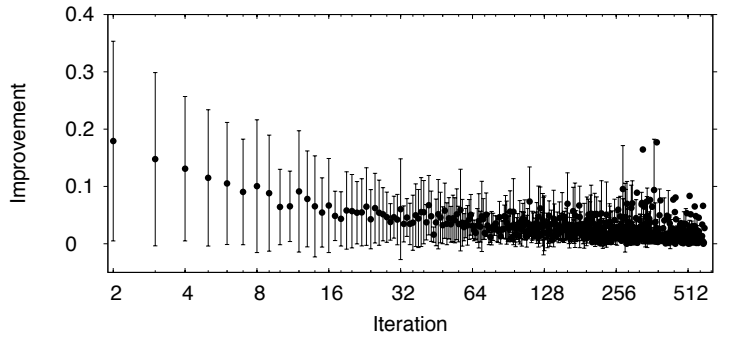
We set the tabu list capacity based on preliminary experiments with $[1, 2, 4, 8, 16, 32]$ percent of the edge count (rounded up with the ceiling function to obtain an integer) as the capacity for all the instances of set A, for all three profit-generation schemes, for five depot vertices selected uniformly at random, with three replicas per depot and a cut-off after 10 iterations of no impromement. For the repetitions of each instance, we assigned six "votes" for the capacity that gave the best result on average, five votes to the second, four votes to the third, etc. We then added up the voted over



(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.

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

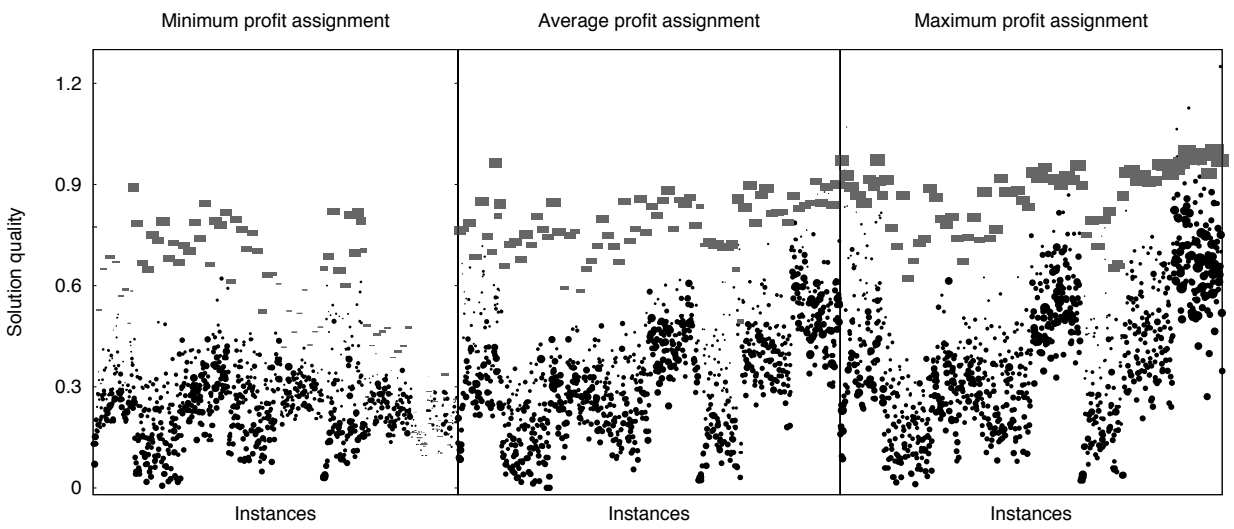


Figure 5: On the vertical axis, the normalized reference values $\hat{\mathcal{R}}$ and the normalized values of the best obtained solutions — cf. Equations (10) and (11), respectively — for the three profit-assignment schemes in set A. The size of each dot is proportional the number of edges included 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.

all instances and chose the value with the highest vote count, which was two percent with a 22% of the total votes (the second-highest votecount was one percent with 21%, but with small graphs these both round up to the same list capacity). We leave more complex tabu search constructions where the entities stored in the tabu list are tour fragments in general instead of just single ages as future work.

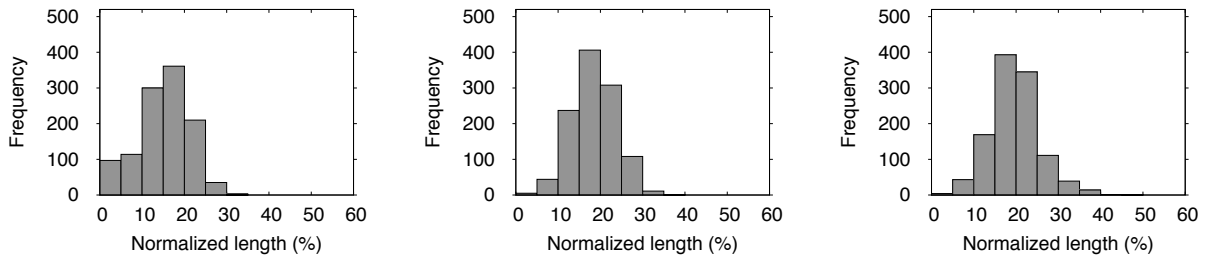
For each execution, we recorded the values of the upper and lower bounds $\mathcal{U}$ and $\mathcal{L}$, respectively, as well as the reference value $\mathcal{R}$, together with the numbers of edges included in the sums of $\mathcal{L}$ and $\mathcal{R}$ — see Equations (7) and (8), respectively. We computed for each input pair (G, d) the total cost of the best closed walk $\mathcal{B}^*$, the length of this best closed walk found, and the time in seconds it took to compute it.

Figure 5 TOBEDONE

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) when all edges are profitable for a single traversal as $p_{vw} \geq \max\{c_{vw}, c_{wv}\}$, 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.

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.

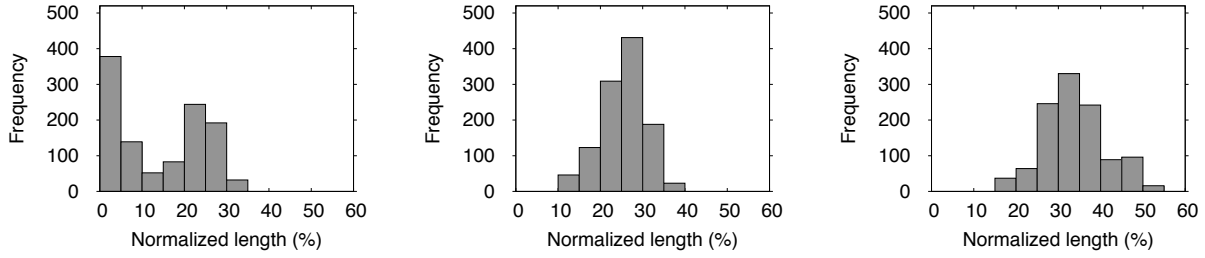
We visualize the execution time of the proposed heuristic on the A set in Figure 7. The horizontal and vertical axes indicate the number of vertices (n) and the number of edges (m), respectively. For the sake of a visual comparison computing times are proportional to dot diameters. Each of the three profit-assignment schemes is drawn separately.



(a) Minimum profit assignment, normalized length of the best solution.

(b) Average profit assignment, normalized length of the best solution.

(c) Maximum profit assignment, normalized length of the best solution.



(d) Minimum profit assignment, normalized length of the reference value.

(e) Average profit assignment, normalized length of the reference value.

(f) Maximum profit assignment, normalized length of the reference value.

Figure 6: Histogram of the normalized lengths of closed walks, using each vertex in turn as depot.

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

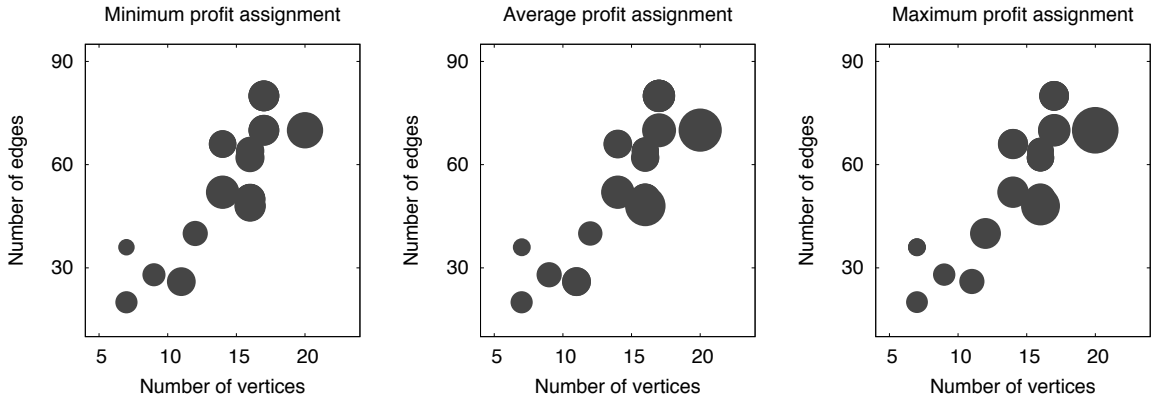


Figure 7: The maximum runtime (represented by the diameter of the dot) of the heuristic within set A with respect to the number of vertices (n) and edges (m).

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. As these graphs require higher runtimes per execution, instead of using all vertices as depots, we used a uniform random samples of five depots per instance. TOBEDONE

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 the upper bound and concentrated around the value $\mathcal{L} + \frac{1}{3}(\mathcal{U} - \mathcal{L})$. In the vast majority of the 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 obtained solutions, as was the case with the instances in set A in Figure 5.

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

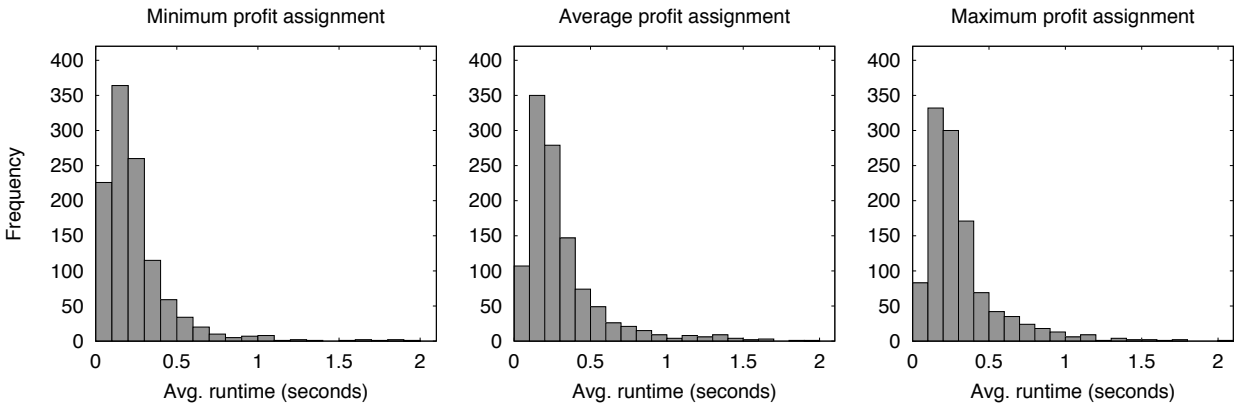
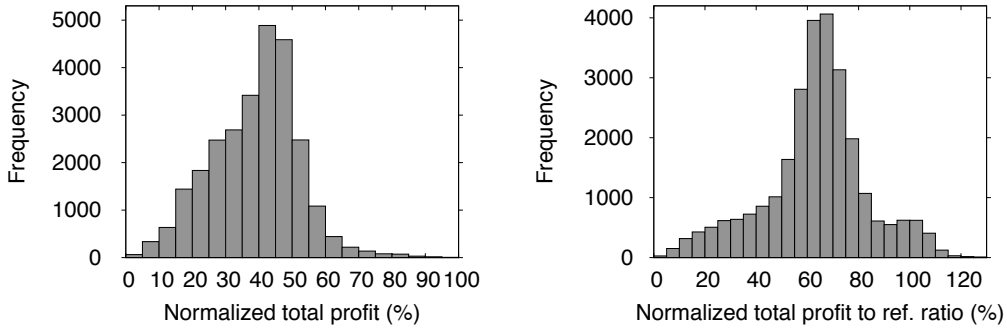
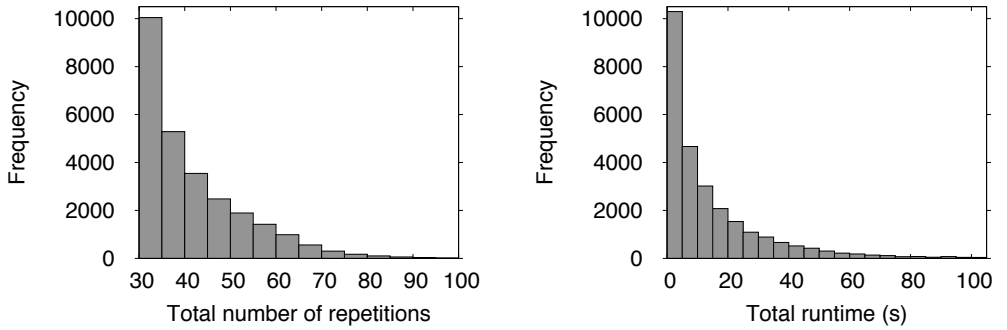


Figure 8: Histograms of the individual runtimes over all executions of the heuristic for the three profit-assignment schemes.



(a) The best normalized profit $100 \times \hat{\mathcal{B}}$.

(b) The ratio $100 \times \hat{\mathcal{B}}/\hat{\mathcal{R}}$.



(c) Iterations per execution.

(d) Total runtimes in seconds.

Figure 9: Histograms of normalized total profits (absolute and relative), numbers of iterations, and runtimes for set B.

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 $\mathcal{O}\left(n^3\right)$ and $\mathcal{O}\left(m^2\right)$ for these instances. In general, $m \in \mathcal{O}\left(n^2\right)$ 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 ≈ 0.15. This explains why we have observed in our experiments $\mathcal{O}\left(m^2\right) \sim \mathcal{O}\left(n^3\right)$.

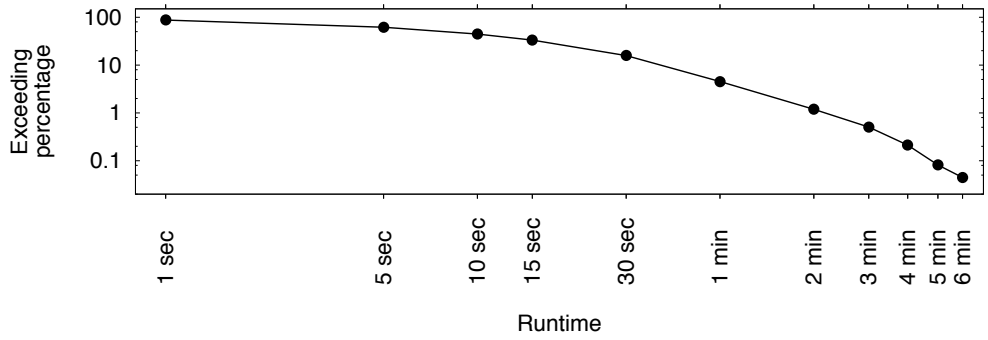
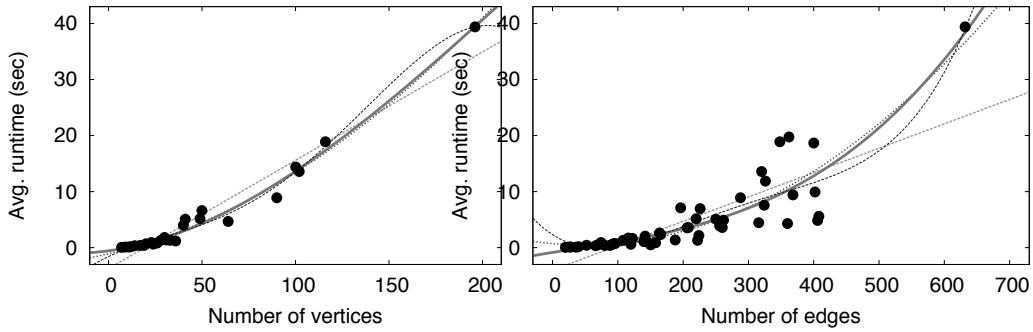
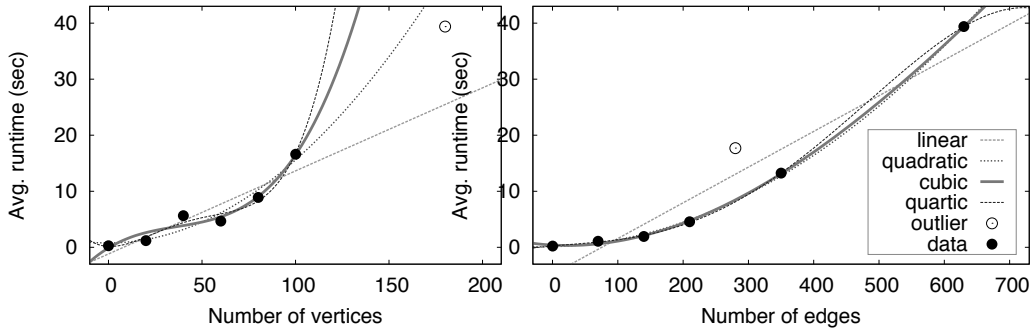


Figure 10: Percentage of instances that exceed a given runtime for instances in set B.



(a) Number of vertices n versus the average runtime; raw data.

(b) Number of edges m versus the average runtime; raw data.



(c) Number of vertices n versus the average runtime; binned data.

(d) Number of edges m versus the average runtime; binned data.

Figure 11: The average runtimes plotted against n and m for set B, with and without binning. In each plot, four polynomial fits are drawn with lines. The same legend applies to all four plots.

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.

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 $\mathcal{B}^*$ obtained for that pair and computed for any other solution value $\mathcal{B}$ that was obtained for that same pairs *tolerance* of how far that value was from $\mathcal{B}^*$:

$$\rho = \frac{\mathcal{B}^* - \mathcal{B}}{\mathcal{B}^*}. \tag{24}$$

Note that there is no limit as such to the values that $\mathcal{B}$ 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 profits are floating-point numbers and the profits were generated pseudo-randomly at uniform 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 κ from the best solution.

We used values for κ 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

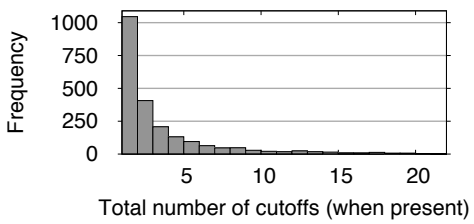


Figure 12: Histogram of the number of cutoffs per execution, when at least one cutoff was present (that is, in 8.6% of the executions).

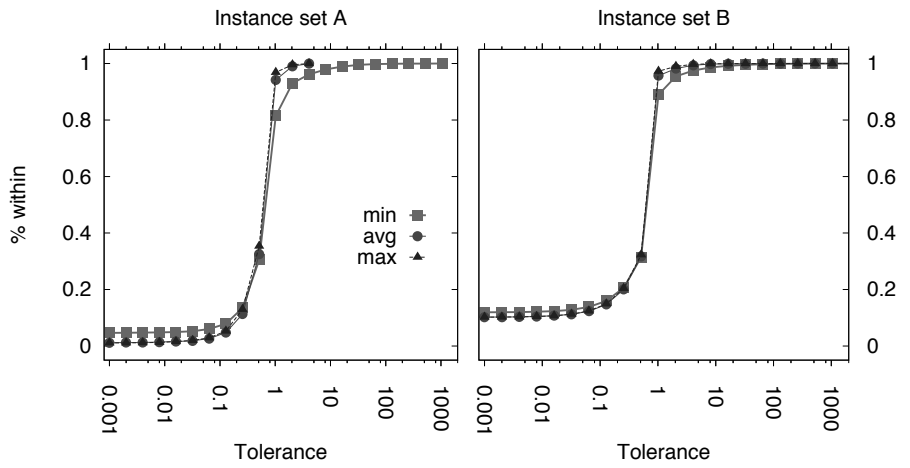


Figure 13: On the horizontal axis, the tolerance level ρ, and on the vertical axis, the percentage of solutions that achieved that tolerance.

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.

Note that presently the heuristic returns a closed walk upon returning to the depot. The optimal solution could very well consist in several visits to the depot, which is easily incorporated to the proposed heuristic by storing all the profitable walks and then attempting combinations of those that had a positive total profit. As the number of possible ways to combine closed walks is exponential, the cardinalities of the differences of the sets of the visited vertices or the included edges could be employed to find tours that traverse largely distinct sections of the input graph and hence would incur little loss of benefit when combined into a single walk with multiple visits to the depot. We leave this combination step to future work in order to keep the presented heuristic simpler.

In future work, we wish to compare the results obtained by the proposed heuristic to exact results on small graphs as well as to a modification of the method proposed by Palma [2011] that is based on tabu search but for a non-windy variant of the problem. We are also interested in a variant of the problem where the depot is not fixed but the selection of the depot is part of the problem formulation, as our experiments revealed that the choice of depot has a powerful impact in the solutions found.

Also parallel 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

J. Aráoz, E. Fernández, and C. Zoltan. Privatized rural postman problems. *Comput. Oper. Res.*, 33(12):3432–3449, 2006.

J. Aráoz, E. Fernández, and C. Franquesa. The clustered prize-collecting arc-routing problem. *Transport. Sci.*, 43(3):287–300, 2009a.

J. Aráoz, E. Fernández, and O. Meza. Solving the prize-collecting rural postman problem. *Eur. J. Oper. Res.*, 196(3):886–896, 2009b.

J. Aráoz, E. Fernández, and C. Franquesa. GRASP and path relinking for the clustered prize-collecting arc routing problem. *J. Heuristics*, 19(2):343–371, 2013.

C. Arbib, M. Servilio, C. Archetti, and M.G. Speranza. The directed profitable location rural postman problem. *Eur. J. Oper. Res.*, 236(3):811–819, 2014.

C. Archetti and M. G. Speranza. Arc routing problems with profits. In Á. Corberáan and G. Laporte, editors, *Arc Routing: Problems, Methods, and Applications*, MOS-SIAM Series on Optimization, chapter 12. SIAM, Philadelphia, 2014. in press.

C. Archetti, D. Feillet, A. Hertz, and M.G. Speranza. The undirected capacitated arc routing problem with profits. *Comput. Oper. Res.*, 37(11):1860–1869, 2010.

E. Benavent, Á. Corberán, G. Desaulniers, F. Lessard, I. Plana, and J.M. Sanchis. A branch-price-and-cut algorithm for the min-max k-vehicle windy rural postman problem. *Networks*, 63(1): 34–45, 2014.

A. Colorni, M. Dorigo, and V. Maniezzo. Distributed optimization by ant colonies. In F.J. Varela and P. Bourgine, editors, *Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, pages 134–142, Cambridge, 1992. The MIT Press.

Á. Corberán, I. Plana, and J.M. Sanchis. The windy general routing polyhedron: A global view of many known arc routing polyhedra. *SIAM J. Discrete Math.*, 22(2):606–628, 2008.

Á. Corberán, E. Fernández, C. Franquesa, and J.M. Sanchis. The windy clustered prize-collecting arc routing problem. *Transport. Sci.*, 45(3):317–334, 2011.

Q. Ding, X. Hu, L. Sun, and Y. Wang. An improved ant colony optimization and its application to vehicle routing problem with time windows. *Neurocomputing*, 98:101–107, December 2012.

Karl F. Doerner, Richard F. Hartl, Vittorio Maniezzo, and Marc Reimann. Applying ant colony optimization to the capacitated arc routing problem. In Marco Dorigo, Mauro Birattari, Christian Blum, Luca Maria Gambardella, Francesco Mondada, and Thomas Stützle, editors, *Ant Colony Optimization and Swarm Intelligence*, volume 3172 of *Lecture Notes in Computer Science*, pages 420–421. Springer, Berlin / Heidelberg, Germany, 2004.

M. Dorigo and M. Birattari. Ant colony optimization. In C. Sammut and G.I. Webb, editors, *Encyclopedia of Machine Learning*, pages 36–39. Springer, New York, 2010.

M. Dorigo and C. Blum. Ant colony optimization theory: A survey. *Theor. Comput. Sci.*, 344 (2–3):243–278, November 2005.

D. Feillet, P. Dejax, and M. Gendrau. Traveling salesman problems with profits. *Transport. Sci.*, 39(2):188–205, 2005.

C. Franquesa. *The Clustered Prize-collecting Arc Routing Problem*. PhD thesis, Technical University of Catalonia, Barcelona, 2008.

K.V. Narasimha, E. Kivelevitch, B. Sharma, and M. Kumar. An ant colony optimization technique for solving min-max multi-depot vehicle routing problem. *Swarm Evol. Comput.*, 13:63–73, December 2013.

Guillermo Palma. A tabu search heuristic for the prize-collecting rural postman problem. *Electronic Notes in Theoretical Computer Science*, 281:85–100, December 2011.

M. Reed, A. Yiannakou, and R. Evering. An ant colony algorithm for the multi-compartment vehicle routing problem. *Appl. Soft. Comput.*, 15:169–176, February 2014.

Luís Santos, Jo ao Coutinho-Rodrigues, and John R. Current. An improved ant colony optimization based algorithm for the capacitated arc routing problem. *Transportation Research Part B: Methodological*, 44(2):246–266, February 2010.

C.-J. Ting and C.-H. Chen. A multiple ant colony optimization algorithm for the capacitated location routing problem. *Int. J. Prod. Econ.*, 141(1):34–44, 2013.