A Genetic Algorithm for the Undirected Rural Postman Problem

Maria do Rosário Moreira[1] José Soeiro Ferreira[1,2]

[1] INESC Porto – Instituto de Engenharia de Sistemas e de Computadores do Porto
[2] FEUP – Faculdade de Engenharia da Universidade do Porto
Emails: mmoreira@inescporto.pt, jsoeiro@inescpoprto.pt

ABSTRACT

This paper introduces a new method to deal with the Undirected Rural Postman Problem (RPP), based on Genetic Algorithms. The RPP is a particular Arc Routing Problem which consists of determining a minimum cost circuit on a graph so that it is possible to traverse a given subset of required edges.
The RPP is known to be an NP-hard problem and it has some interesting real-life applications. The authors are aware of only one publication using Genetic Algorithms alone in this type of problems.
The new method – the RPGA algorithm – will be described in detail and tested with different groups of problems, the results being compared with those gathered from other publications. A comparison is also made with several crossovers frequently used in TSP problems. A new case based on the city of Porto is also solved.
The simplicity of RPGA and the computational experiments suggest that it can be a good choice to solve (undirected) RPP, in a short period of time.

KEYWORDS: Arc Routing Problems, Rural Postman Problem, Genetic Algorithms

1. The Rural Postman Problem

This paper introduces a new method – the RPGA algorithm – based on Genetic Algorithms to solve the Undirected Rural Postman Problem (RPP). The RPP is a particular Arc Routing Problem, with many applications. Perhaps the earliest reference to ARP is related to Euler, known as the Konnisberg Bridge Problem, Euler [7]. The RPP is more general than the Chinese Postman Problem, which has as objective finding the minimum cost circuit traversing, at least once, all the edges of a given graph. It may be solved in $O(|v^3|)$ time, Edmonds et al [6]. The RPP considers two types of edges, required and non-required edges and may be defined as follows:
Let $G = (V, E)$ be an undirected and weighted graph with a vertex set $V = \{v_1, v_2, ...v_m\}$, an edge set $E = \{e_1, e_2, ...e_n\}$, a subset $E_r \subseteq E$ of "required" edges (they must be traversed) and let c_i be the cost of edge e_i. The objective of the RPP is to find a minimum cost circuit so that each edge of E_r is traversed at least once. The RPP was first introduced by Orloff in Orloff [16].
A large number of real-world problems are related to RPP: routing of newspapers, post delivery, the collection of household refuse or parking meter coins, street sweepers, school buses, and the inspection of electric power or oil or gas pipelines, see Pearn (et al) [18]. A new application in the area of cutting tool path determination may be found in Moreira et al [15]. The RPP is classified as NP-hard (Lenstra, et al [14]) what may be a motivation to use Metaheuristics as an alternative approach to 'exact methods'. There are some publications related to the RPP. Frederickson's heuristic, Frederickson (et al) [8], is similar to the heuristic of Christofides for the Travelling Salesman Problem, operating by adding artificial edges to the subgraph induced by E_r in a way that yields a connected Eulerian graph. Christofides (et al) [2] developed a heuristic algorithm consisting of three stages: firstly, transforming the subgraph containing the required edges in a complete graph; secondly, applying a minimal spanning tree algorithm and, finally, by means of the minimum-cost matching algorithm, obtaining a Eulerian graph. Corberán (et al) [3] developed a branch and cut algorithm; Córdoba et al [4] introduced a heuristic algorithm for the RPP based on a Monte Carlo method. Later, Hertz (et al) [11] set up a family of local search heuristics. Rodrigues (et al) [19] proposed a Memetic Algorithm. Baldoquín (et al) [1] developed a hybrid approach based on GRASP, and Genetic Algorithms and Peña [17] used Simulated Annealing, GRASP and Genetic Algorithms. Groves (et al) [10] presents a local search framework. All these methods used the same 26 instances, with the exception Groves et al.[10], who dealt with new large instances. Moreira (et al) [15] proposed an algorithm for a dynamic RPP. The authors are aware of a single publication using Genetic Algorithms only, due to Kang (et al) [13]: the original graph is transformed into a Hamiltonian circuit and the PMX crossover is applied; the results are for little dimension problems. Baldoquín (et al) [1] and Peña [17] used the Genetic Algorithms, but together with others heuristics.

A very brief introduction to Genetic Algorithms is presented in Section 2. In Section 3, RPGA – a new method based on Genetic Algorithms – will be proposed to solve the RPP. The computational results and the evaluation are included in Section 4. RPGA was applied to different groups of problems, some new and others used in referred publications. Finally, in Section 5, it is concluded that the simplicity of the method and the results achieved confirm RPGA as a potential method to deal with RPP.

2. Genetic Algorithms

Genetic algorithms (GA) were firstly proposed by Holland [12] in his book *Adaptation in Natural and Artificial Systems* whose ideas were applied and expanded by Goldberg [9], who simulated biological processes. GA are based on the principles of the Darwin Theory and, basically, they may be described as follows.
The algorithm begins with a set of solutions, encoded by chromosomes, that is called the initial population. Next, a value (fitness) is attributed to each chromosome. Then the best ones (according to their fitness) are selected: these are the parents. To create a new generation, parents will be crossing with a probability p_c to generate new chromosomes: the children. In the new population, the worst ones are eliminated and the process repeats itself for N generations, where N is a previously defined number. GA's main strength comes from the *Crossover* operation: an exchange of part of the information, the *genes*, contained in the chromosomes of the selected parents. When the chromosomes are not submitted to crossover, they remain unmodified. Various kinds of crossovers have been suggested, see Goldberg [9]. *Mutation* involves the modification of the value of some genes of a chromosome with a p_m probability (the mutation probability). This process re-introduces lost or unexplored genetic material back into the population, with the intention of preventing premature convergence.
The process is repeated until a stopping condition is verified, usually the number of generations. If all goes well throughout this entire procedure, the initial population (of no special quality) will improve as parents are replaced by better and better children. In this paper, a particular adaptation of the idea of GA is implemented: the algorithm will consider the situation of the generation of a son by only one father.

3. RPGA – A Genetic algorithm for the Undirected Rural Postman Problem

This section describes the new algorithm RPGA to solve the (undirected) RPP. The algorithm starts by constructing an auxiliary Hamiltonian graph $G^H = (V^H, E^H)$ as follows:

1. For each $e_i \in E_r$ create a vertex $v_i^H \in V^H$
2. For each pair $v_i^H, v_j^H \in V^H (i \neq j)$ create an edge $\left(v_i^H, v_j^H\right) \in E^H$, whose cost for each edge is calculated using a shortest path algorithm. The resulting graph is a completely connected graph.

This procedure is similar to the one in Kang (et al) [13]. The RPP problem is then transformed into a TSP.

3.1. Encoding

The process for encoding solutions is often a critical aspect in solving a problem using GA. Holland worked with a binary system, but new representations appeared shortly afterwards.
It was decided to follow an ordinary and close to the problem encoding: the genes of each chromosome should be represented by the vertices in V^H. In this way, if the chromosome is (1, 2, 4, 5, 3), the path starts in vertex 1, and continues through vertex 2, 4, 5 and 3.

3.2. Fitness Function

The objective of the problem is to minimize the total cost

$$C = \sum_{i=1}^{n} (c(e_i) + h_i) \qquad (*)$$

where $n = |E^r|$, $c(e_i)$ is the cost of edge e_i ($e_i \in E_r$), and h_i is the minimum cost from the end vertex of $e_i \in E_r$ to the start vertex of $e_{i+1} \in E_r$. If i = n, h_i is the minimum cost from the end vertex of $e_n \in E_r$ to the start vertex of $e_1 \in E_r$. These calculations are based on the path under construction waht will be clarified later with an example.
The fitness function f and the probability of selecting one chromosome k are defined as

$$f = \frac{1}{C} \qquad \text{and} \qquad p[k] = \frac{f_k}{\sum_{j=0}^{l} f_j}$$

where f_k is the fitness of the chromosome k and l+1 is the size of the population.

3.3. Initial Population

The initial population is randomly generated. The choice of the size of the population is also a relevant issue: if the population is too small, the GA may converge too quickly; if it is too large, the process may take a long time. After some computational experiments 200 individuals were considered for the medium/large problems and 100 for the small problems. Only the 80% of the population's best part was taken into account, for each population. Also, the best chromosome in the population was immediately stored for the next generation.

3.4. Child Construction

RPGA has no crossover operation. A child is obtained from a single father, through the following operation.

The first vertex (in V^H) in which h_i (in *) is non zero is selected and a copy of all the genes until this position in the child's chromosome is made. Then to complete the chromosome:

1) The algorithm looks for all vertexes (not yet selected) such that $h_i \neq 0$ until the last one, (if there is more than one vertex available, one of them will be randomly selected). When a vertex such that $h_i \neq 0$ does not to exist, it goes to 2).
2) The algorithm connects the last vertex as follows:
    a. In 99% of the situations, it looks for the vertex whose h_i is minimum;
    b. In 1% of the situations, it carries out a random selection between the vertexes that were not selected;
3) If the chromosome is complete it stops. Else it goes to 1).

After selecting the vertex in the auxiliary graph G^H, the algorithm determines the corresponding edge in the original graph. The direction of this edge is chosen so as to minimize the cost.

Adaptive construction probabilities p_c were applied, following the work of Srinivas (et al) [20] and after carrying out the convenient adaptations

$$p_c = \begin{cases} \dfrac{f_{\max} - f}{f_{\max} - \overline{f}} & \Rightarrow f \geq q_1 \\ 1 & \Rightarrow f < q_1 \end{cases}$$

$f_{\max} \rightarrow$ The maximum fitness in the old population

$\overline{f} \rightarrow$ The average fitness in the old population

$f \rightarrow$ The fitness of the parent who is a candidate to be progenitor

$q_1 \rightarrow$ The first quartile

Srinivas (et al) [20] use the average instead of the first quartile and this probability is used for the crossover.

Next example is intended to illustrate the essential features of RPGA. The data is taken from problem 1 in Christofides (et al) [2], and a graph is represented in Fig. 1.

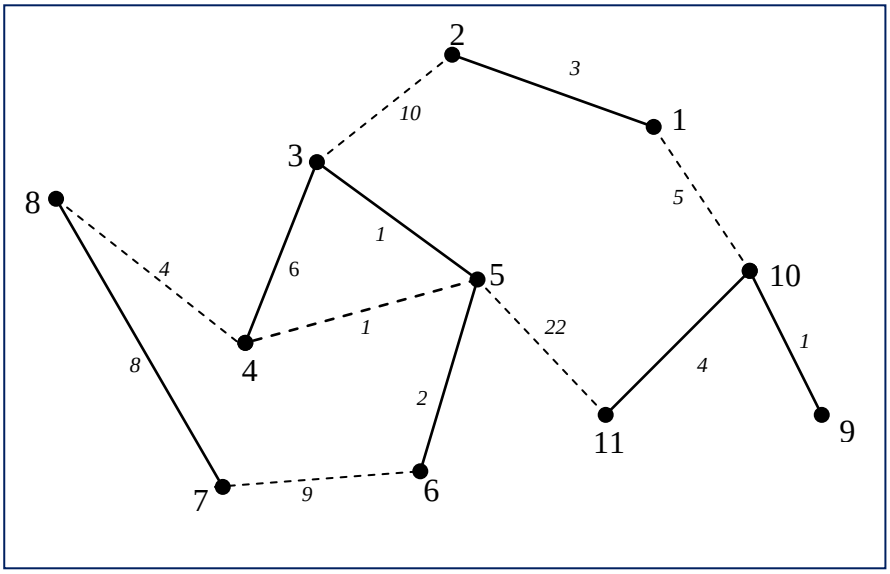

Fig.1. A graph from problem 1 in Christofides (et al) [2]

Table 1 has the necessary information about the required edges: the extreme vertexes and the cost in the original graph (in italic), as well as the corresponding vertexes V^H in G^H.

Required edges in Graph G	*Extreme Vertexes*	*1,2*	*3,4*	*3,5*	*5,6*	*7,8*	*9,10*	*10,11*
	Cost	*3*	*6*	*1*	*2*	*8*	*1*	*4*
v^H		1	2	3	4	5	6	7

Table 1. Date of problem 1 in Christofides (et al) [2]

Table 2 contains the minimum costs between vertexes in G.

Vertexes	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
1		3	13	15	14	16	25	19	6	5	9
2			10	12	11	13	22	16	9	8	12
3				2	1	3	12	6	19	18	22
4					1	3	12	4	21	20	23
5						2	11	5	20	19	22
6							9	7	22	21	24
7								8	31	30	33
8									25	24	27
9										1	5
10											4
11											

Table 2. The minimum costs between the vertexes in G

The corresponding edges in the original graph are in italics.
Figures illustrate the path that is traversed in the original graph.

Suppose the parent is:

P = (3 4 2 5 1 6 7)

(3,5)-(5,6)-(3,4)-(7,8)-(2,1)-(10,9)-(10,11)

It is possible to observe that vertexes 3 and 4 are such that $h_1 = 0$, while $h_2 \neq 0$. Therefore, the algorithm only copies 3 and 4 into the child chromosome.
So the child will start with

C = (3 4 - - - - -),

which corresponds to edges *(3,5)-(5,6)* in the original graph.
The current path is represented (in bold) in Fig. 2:

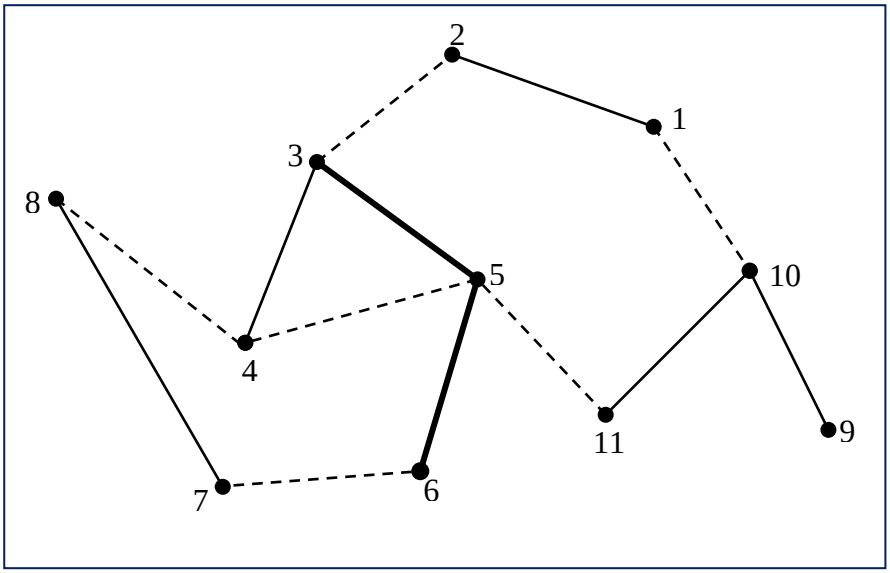

Fig. 2. The current path in the original graph

To complete the child, in each step the algorithm stores the set (A) of the vertexes in the transformed graph that haven't been selected yet. The current set is:

$$A = \{1,2,5,6,7\}$$

Next, the algorithm tries to find, in set A, a vertex with h_2 =0 (if there is more than one, one of the possible vertexes will be randomly selected). In this example, all the vertexes such that $h_2 = 0$ have already been selected. The algorithm goes to step 2. Let us suppose that hypothesis a. applies.

V^H	h_2 (see table 2)
1	$\min\{c(6,1); c(6,2)\} = c(6,2) = 13$
2	$\min\{c(6,3); c(6,4)\} = c(6,3) = c(6,4) = 3$
5	$\min\{c(6,7); c(6,8)\} = c(6,8) = 7$
6	$\min\{c(6,9); c(6,10)\} = c(6,10) = 21$
7	$\min\{c(6,10); c(6,11)\} = c(6,10) = 21$

Select $V^H = 2$.

The direction could be (3,4) or (4,3) (The same cost)

The child becomes

C=(3 4 2 - - - -)
(3,5)-(5,6)-(3,4)

and

$$A = \{1,5,6,7\}.$$

Vertex 2 has no vertex such that $h_3 = 0$. Considering again hypothesis a. in step 2:

V^H	h_3
1	$\min\{c(3,1); c(3,2); c(4,1); c(4,2)\} = c(3,2) = 10$
5	$\min\{c(3,7); c(3,8); c(4,7); c(4,8)\} = c(4,8) = 4$
6	$\min\{c(3,9); c(3,10); c(4,9); c(4,10)\} = c(3,10) = 18$
7	$\min\{c(3,10); c(3,11); c(4,10); c(4,11)\} = c(3,10) = 18$

Select $V^H = 5$.

The direction for v^H=2 is (3,4) and the direction for v^H=5 is (8,7)

The child becomes

C=(3 4 2 5- - -)
(3,5)-(5,6)-(3,4)-(8,7)

and

$$A = \{1,6,7\}.$$

Vertex 3 has no vertex such that $h_4 = 0$. Assuming this time hypothesis b. in step 2, the vertex v^H=1 will be selected.

C=(3 4 2 5 1- -)
(3,5)-(5,6)-(3,4)-(8,7)-(2,1)

(Notice that the direction is (2,1) and not (1,2), because C(1,7)=25 and c(2,7)=22) and

$$A = \{6,7\}.$$

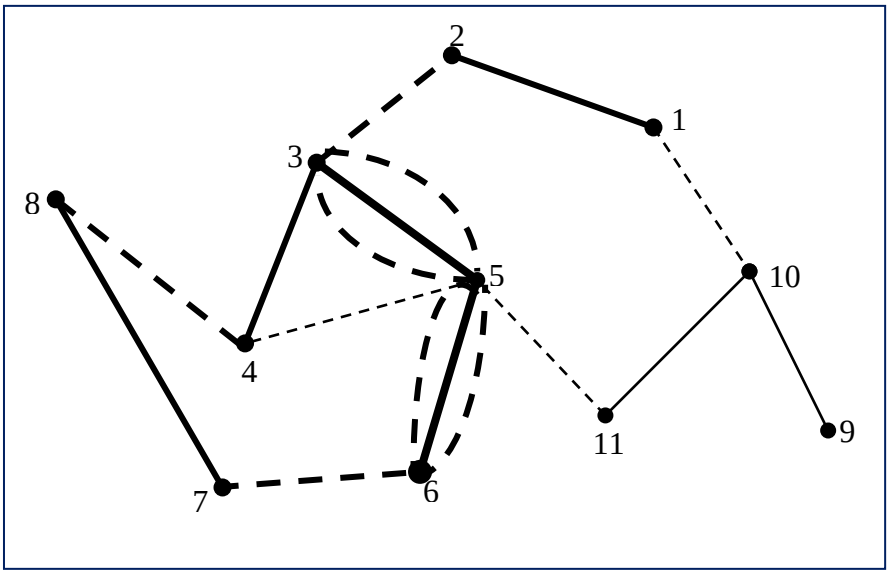

Fig. 3. The path already found in the original graph is shown in bold

Now h_5 is the same, independently of choosing vertex 7 or vertex 6. Suppose that the algorithm randomly vertex 7 and finally the vertex 6:

C=(3, 4, 2 5, 1, 7, 6)

(3,5)-(5,6)-(3,4)-(8,7)-(2,1)-(10,11)-(10,9))

The final path is shown in next graph:

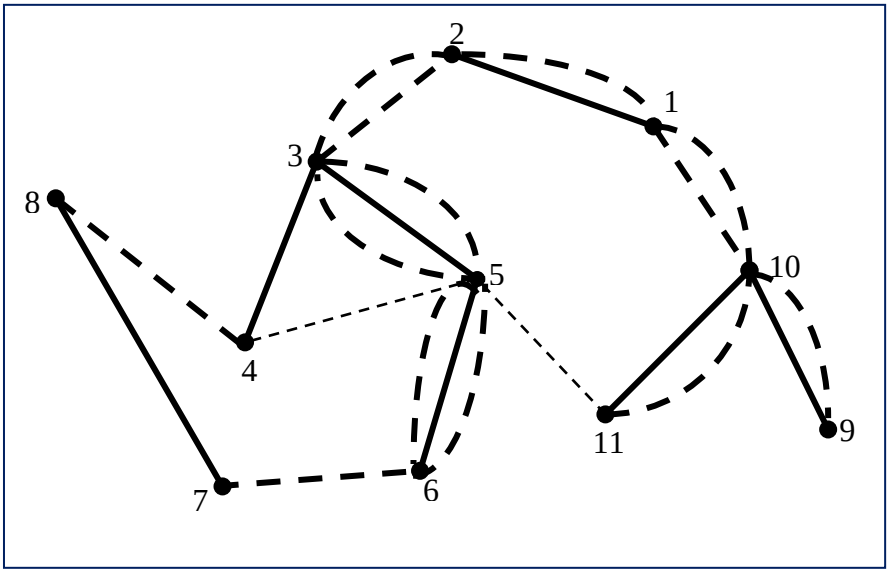

Fig. 4. The final path is shown in bold in the original graph

3.5. Mutation

Mutation operation is a reciprocal exchange. If the original chromosome is (1 2 3 **5** 7 8 9 **6** 4) two positions are selected randomly (position 4 and 8). The transformed chromosome will be (1 2 3 **6** 7 8 9 **5** 4).

The vertex $v_{i+1} \in V^H$ whose h_i is maximum has a probability 1 of being permuted, and when h_i is null this probability is zero.
Adaptive mutation probabilities p_m were applied:

$$p_m = \begin{cases} 0{,}5\dfrac{f_{\max} - f}{f_{\max} - \overline{f}} & \Rightarrow f \geq \overline{f} \\ 0{,}5 & \Rightarrow f < \overline{f} \end{cases}$$

$f_{max} \rightarrow$ The maximum fitness in the old population
$\overline{f} \rightarrow$ The average fitness in the old population
$f \rightarrow$ The fitness of the parent, which is a candidate to be mutated.

4. Computational Results

This section includes extensive computational results with RPGA and other methods. The experiments were organised in three parts: Parts A and B include the same 26 problems taken from the literature and Part C takes account of a new problem based on the city of Porto.
Part B illustrates a comparison of results between RPGA and two different versions of this algorithm, the outcomes from the introduction of two frequently used crossovers for the TSP: the PMX (partially mapped crossover) defined by Goldberg [9] and the OX (order crossover) proposed by Davis [5]. PMX builds children by choosing a subsequence of a tour from one parent and preserving the order of as many genes as possible from the other parent. OX builds children by choosing a subsequence of a tour from one parent and preserving the relative order of genes from the other parent. The difference between these crossovers is crucial as it explains why the OX performs much better than the PMX for the RPP.
The following notation is employed:

#V – number of vertex
#RE – number of required edges
#NRE – number of non-required edges
OV – known optimal value (in the publications)
BVGA – best value obtained with RPGA
R=OV/BGA – ratio between the known optimal value and the best value obtained with RPGA OV/BGA
T - computational time (in seconds)

The computer used has an Intel Pentium IV 3.2. GHz processor and 512 Mb RAM.

Part A
Table 3 presents 24 problems (P1 to P24) found in Christofides (et al) [2] and 2 problems (P25, P26) found in Corberán (et al) [3], and includes the corresponding results obtained with RPGA and with the following methods:

- the heuristics described in Christofides (et al) [2];
- the Monte Carlo method due to Córdoba (et al) [4];
- the hybrid heuristic depicted in Peña [17];
- and the heuristic due to Groves (et al) [10].

Problem	#V	#RE	#NRE	OV	BVGA	R	T	Christ. Heur	M.C. Heur	Peña Heur	Groves Heur
P1	11	7	6	76	76	1	<1	76	76	76	76
P2	14	12	21	152	152	1	<1	**164**	**163**	**163**	152
P3	28	26	31	102	102	1	<1	102	102	102	**103**
P4	17	22	13	84	84	1	<1	84	**86**	84	84
P5	20	16	19	124	124	1	<1	**135**	**129**	**129**	124
P6	24	20	26	102	102	1	<1	**107**	102	102	102
P7	23	24	23	130	130	1	<1	130	130	130	130
P8	17	24	16	122	122	1	<1	122	122	122	122
P9	14	14	12	83	83	1	<1	**84**	83	83	83
P10	12	10	10	80	80	1	<1	80	**84**	80	80
P11	9	7	7	23	23	1	<1	23	23	23	23
P12	7	5	13	19	19	1	<1	**22**	**21**	**21**	19
P13	7	4	6	35	35	1	<1	**38**	**38**	**38**	35
P14	28	31	48	202	202	1	<1	**212**	**209**	**209**	202
P15	26	19	18	441	441	1	<1	**445**	**445**	**445**	441
P16	31	34	60	203	203	1	2	203	203	203	203
P17	19	17	27	112	112	1	<1	**116**	112	112	112
P18	23	16	21	146	148	0,99	2	148	148	148	___
P19	33	29	25	257	257	1	<1	**280**	**263**	**263**	**266**
P20	50	63	35	398	398	1	6	**400**	**399**	398	**400**
P21	49	67	43	366	366	1	8	**372**	**368**	**366**	**372**
P22	50	74	110	621	621	1	10	**632**	621	**621**	**622**
P23	50	78	80	475	475	1	4	**480**	**489**	**480**	**477**
P24	41	55	70	405	405	1	2	**411**	405	405	405
P25	102	99	61	10599	11198	0,95	10	----	10784	10612	10599
P26	90	88	56	8629	8629	1	15	----	**8721**	8629	8629

Table 3. Characterization and results of the 26 problems of set A

A first conclusion may be drawn: RPGA is the only algorithm that reaches the optimal solution in 23 problems for the P- Problem. In the case of the two problems P25 and P26, RPGA does not reach the optimal solution for P25 as does Peña heuristic. Groves heuristic reached the optimal solution in both the instances given by Corberán & Sanchis [1994].

Part B

The same set of problems is considered (see Table 4). The comparisons are between RPGA and, as it has been mentioned, two different versions of this algorithm resulting from the use of PMX and OX crossovers.

Problem	OV	BVGA	R	T	PMX	T	OX	T
P1	76	76	1	<1	76	<1	76,0	<1
P2	152	152	1	<1	152	<1	152	<1
P3	102	102	1	<1	**109**	**46**	**107**	**77**
P4	84	84	1	<1	**85**	2	**86**	2
P5	124	124	1	<1	124	<1	124	<1
P6	102	102	1	<1	**104**	<1	102	4
P7	130	130	1	<1	130	57	130	29
P8	122	122	1	<1	122	252	122	<1
P9	83	83	1	<1	83	145	83	1
P10	80	80	1	<1	80	8	80	<1
P11	23	23	1	<1	23	<1	23	<1
P12	19	19	1	<1	19	<1	19	<1
P13	35	35	1	<1	35	<1	35	<1
P14	202	202	1	<1	202	104	202	20
P15	441	441	1	<1	441	2	441	1
P16	203	203	1	2	203	123	203	1
P17	112	112	1	<1	112	97	112	1
P18	146	148	0,99	2	146	72	146	1
P19	257	257	1	<1	**259**	**793**	257	**2**
P20	398	398	1	6	**402**	**596**	**400**	**63**
P21	366	366	1		**376**	**529**	**381**	**15**
P22	621	621	1	10	**627**	**1071**	**625**	**105**
P23	475	475	1	4	**481**	**3801**	**479**	**161**
P24	405	405	1	2	**424**	471	408	102
P25	10599	11198	0,95	10	11963	3963	10727	61
P26	8629	8629	1	15	10294	1740	9503	104

Table 4. Results with RPGA and the versions using PMX and OX crossovers.

The first and main conclusion is that RPGA performs better than the versions using PMX and OX crossovers, both in what concerns quality of the solutions and computational time. The explanation is that the algorithm takes into account the fact that the genes to be permuted are not the same probability and that the edges of the transformed graph do not always have the same cost, as elucidated in Section 3. It was also expected that the crossover OX would prove more efficient than the PMX, since OX takes into consideration the relative position of the genes, while the PMX does not.

Part C

A new problem derived from the street network of the city of Porto is solved with RPGA and with the versions of the algorithm using PMX and OX crossovers.

The graph has a total number of 656 vertexes and 876 edges. The required edges are randomly constructed, with probabilities of 20%, 40%, 60% and 80% respectively for the groups Porto2*, Porto4*, Porto6* and Porto8*.

Since the optimal solutions are not known, Table 5 also includes the total fitness of required edges, represented by ‘rFit’.

The data may be obtained from the authors.

Problems	#V	#RE	#NRE	rFit	BVGA	T	PMX	T	OX	T
Porto 21	656	177	699	2868	8144	61	8734	2829	8303	308
Porto 22	656	182	694	3071	8419	76	8933	923	8733	655
Porto 23	656	171	705	2793	7581	70	7700	6858	7490	455
Porto 41	656	359	517	5749	11218	200	12145	412	11674	680
Porto 42	656	354	522	5762	11325	210	12114	1247	11906	268
Porto 43	656	352	524	5602	11479	230	11953	354	12121	273
Porto 61	656	549	327	9024	14417	421	15562	2415	15526	351
Porto 62	656	525	351	8306	13805	444	14735	585	14609	292
Porto 63	656	537	339	8670	14341	500	15536	910	15336	579
Porto 81	656	703	173	11215	16306	700	17765	3211	17709	1396
Porto 82	656	711	165	11202	16563	721	17913	2305	17824	1045
Porto 83	656	695	181	10981	16394	732	17882	19683	17883	922

Table 5. Results for Porto problem

The results obtained confirm, once again, that RPGA performs better than the alternative versions.

5. Conclusions

The new algorithm (RPGA) used to tackle the Undirected Rural Postman Problem (RPP) was presented.
The paper includes a review of previous works on RPP and, as it has been pointed out, there are almost no publications using Genetic Algorithm (GA).
The RPGA is based on GA, trying to take into account the specificities of the RPP. That is particularly reflected in the encoding, child construction and mutation operations.
The computational experiments confirmed that it works very well when compared to other heuristics known from the literature. A new problem deriving from the city of Porto was also solved.
As a final conclusion, and considering that the implementation of the algorithm is very simple, RPGA is a good choice to deal with the (undirected) RPP.

References

1. Baldoquín M., Ryan, G., Rodríguez, R., Castellini, A., *Un enfoque híbrido basado en metaheurísticas para el Problema del Cartero Rural*, Proceedings of XI CLAIO, Concepción de Chile, Chile, 2002
2. Christofides N., Campos V., Corberán, A., *An algorithm for the Rural Postman Problem*, Technical Report IC.OR.81.5. Imperial College Report, March 1981
3. Corberán, A., Sanches, J., *A polyhedral Approach for the Rural Postman Problem*, European Journal of Operational Research, 79, 95-114, 1994;
4. Córdoba, P., Raffi, L., Sanchis, J., *A Heuristic algorithm based on Monte Carlo methods for the rural postman problem*, Comp. Oper. Res., 25 (12), 107-1106, 1998
5. Davis, L., *Applying Adaptative Algorithm to Epistatic Domains*, Proceedings of the international Joint Conference on artificial intelligence, 162-164, 1985

6. Edmonds, J., Johnson, E., Machings, *Euler Tours and Chinese Postman problem*, Math. Program, 5, 88-124, 1973
7. Euler, L., *The Koningsberg bridges*, Scientific American, 189, 66-70, 1953
8. Frederickson, G., *Approximation Algorithms for some postman problems*, Journal of the Association for Computing Machinery, 26 (3), 538-554, 1979
9.Goldberg, D., *Genetic algorithms in Search, Optimization, and Machine learning*, Addison –Wesley, Reading, Massachusetts, 1989.
10. Groves, G., Vuuren, J., *Efficient heuristics for the Rural Postman Problem*, ISSN 0529-191-X°c 2005, http://www.orssa.org.za: 33–51, 2005
11. Hertz, A., Laporte, G., Hugo, P., *Improvement procedures for the undirect rural postman problem*, INFORMS Journal on computing, 11(1), 53-62, 1999
12. Holland, J., *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Abor, 1975.
13. Kang, M., Han., C., *Solving the Rural Postman Problem using a Genetic Algorithm with a Graph Transformation*, Proceedings of the 1998 ACM symposium on applied computing. ACM Press, 1998
14. Lenstra, J., Rinnoy-Kan, A., *On general routing* problem, Networks, 6, 273-280, 1976
15. Moreira, L., Oliveira J., A. Gomes, Ferreira, J.S., *Heuristics for a Dynamic Rural Postman* Problem, Computer & Operation Research. 34, 2181-3294, 2007
16. Orloff, C., *A fundamental Problem in vehicle routing*, Networks, 4, 35-64, 1974
17. Peña, M., *Heuristics and metaheuristics approaches used to solve the Rural Postman Problem: A Comparative Case Study*, http://www.x-cd.com/eis04/22.pdf , 2005
18. Pearn, W., Wu, T., *Algorithms for the Rural Postman Problem*, Computers Ops. Res, 22, 819-828, 1995
19. Rodrigues, A., Ferreira, J.S., *Solving the Rural Postman Problem by Memetic Algorithms*, MIC 2001-4th Metaheuristics International Conference, Porto, Portugal, 679-683, 2001
20. Srinivas, M., Patnaik, L., *Adaptive Probabilities of Crossover and Mutation in Genetic Algorithms*, IEEE - Transaction on Systems, Man and Cybernetics, 24, 656-667, 1994