Solving the team orienteering problem

Developing a solution tool using a genetic algorithm approach

João Ferreira[1], Artur Quintas[2], José A. Oliveira[1], Guilherme A.B. Pereira[1], and Luis Dias[1]

[1] Centre Algoritmi, Universidade do Minho, Braga, Portugal
joao.aoferreira@gmail.com, {zan,gui,lsd}@dps.uminho.pt
[2] Graduation in Informatics Engineering, Universidade do Minho, Braga, Portugal
a51836@alunos.uminho.pt

Abstract. Nowadays, the collection of separated solid waste for recycling is still an expensive process, specially when performed in large-scale. One main problem resides in fleet-management, since the currently applied strategies usually have low efficiency. The waste collection process can be modelled as a vehicle routing problem, in particular as a Team Orienteering Problem (TOP). In the TOP, a vehicle fleet is assigned to visit a set of customers, while executing optimized routes that maximize total profit and minimize resources needed.
The objective of this work is to optimize the waste collection process while addressing the specific issues around fleet-management. This should be achieved by developing a software tool that implements a genetic algorithm to solve the TOP.
We were able to accomplish the proposed task, as our computational tests have produced some challenging results in comparison to previous work around this subject of study. Specifically, our results attained 60% of the best known scores in a selection of 24 TOP benchmark instances, with an average error of 18.7 in the remaining instances.
The usage of a genetic algorithm to solve the TOP proved to be an efficient method by outputting good results in an acceptable time.

Keywords: Routing Problems, Team Orienteering Problem, Optimization, Metaheuristics, Genetic Algorithm

1 Introduction

In nature, there is no other living creature that endangers more the environment than us humans. Our way of living and eating habits represent the major source of solid waste production, along with the level of technology consumption. In order to manage and control this continuous waste generation, some decisions were taken. Giving a better and useful utilization to waste materials (recycling) is one of them, along with the corresponding reduction of the total amount of waste produced. In this case, waste separation is critical in order to generate a special

WSC17 Online version
2012 Online Conference on
Soft Computing in Industrial Applications
Anywhere on Earth, December 10-21, 2012

stream of solid waste aside from the common waste. Therefore, a new collection system was developed and different collection points were made available to the population. There are companies specially dedicated to the separated waste collection for recycling, and that task is usually performed using a vehicle fleet, with fixed routes and schedules.

The real problem lies, not on designing an easily accessible network of collection points, but yet on the development of efficient methods for performing waste collection, where constant resource management is vital. One issue that often arises is how to find a way to perform waste collection with a limited fleet of vehicles while obtaining the highest possible profit. An approximation to this issue is the well-known Vehicle Routing Problem (VRP), described in the literature as the problem of designing the least-costly routes from a depot to a set of customers of known demand, where each customer is visited exactly once, without violating capacity constraints and aiming to minimize the number of vehicles required and the total distance travelled. However, the majority of real-world applications require more flexible systems that may lead to the selection of customers. The Team Orienteering Problem (TOP) models can be applied to solve that issue. In the TOP, each customer has an associated profit, and the routes have maximum durations or distances. The choice of customers is made by balancing their profits and their contributions for the route duration or distance. The objective is to maximize the total reward collected in all routes while satisfying the time or distance limit.

The TOP is a fairly recent concept, first suggested by Butt and Cavalier (1994) under the name of Multiple Tour Maximum Collection Problem. Later, Chao et al. (1996) formally introduced the problem and designed one of the most frequently used sets of benchmark instances. In 2006, Archetti et al. achieved many of the currently best-known solutions for the TOP instances by presenting two versions of Tabu Search, along with two metaheuristics based on Variable Neighbourhood Search (VNS). Other competitive approaches were carried out by Ke et al. (2008), with two Ant Colony Optimization (ACO) variations; Vansteenwegen et al. (2009a), with a VNS-based heuristic; and more recently, Souffriau et al. (2010) designed two variants of Greedy Randomized Adaptive Search Procedure with Path Relinking.

The work presented in this paper is part of a series of experiments integrated in the R&D project named Genetic Algorithm for Team Orienteering Problem (GATOP), which was approved by the Portuguese Foundation for Science and Technology (FCT). It involves five combined tasks to accomplish the desired goal which is the development of a more complete and efficient solution for several real-life multi-level Vehicle Routing Problems, with emphasis on the waste collection management. Within the GATOP project, the main task is to solve the TOP, and the development of heuristic solutions based on a genetic algorithm (GA) is suggested. The simplicity of a GA in modelling more complex problems and its easy integration with other optimization methods were the factors considered for its choice. We believe it can be applied to solve the TOP, since it was also used for the Orienteering Problem by Tasgetiren (2002).

In this work we propose to solve medium-to-large-scale TOP instances considering a time constraint. We intend to verify whether it is possible to develop a method, based on a GA, that optimizes the TOP by achieving equal or better results as presented in previous studies.

2 Problem Formulation

The aim of the present study is to solve the TOP, which means to develop a method that determines P paths which start in the same location and have the same destination, in order to maximize the total profit made in each path, while respecting a time constraint. Then, the generated paths are assigned to a limited vehicle fleet, usually one path to each vehicle available.

Following the mathematical formulation suggested by Ke et al. (2008), the objective function of the TOP is presented in Equation 1, where n is the total number of customers, m is the number of vehicles available, the value y shows if customer i is visited or not by a vehicle k, and finally, r is the reward associated to a certain customer i. The objective function consists of finding m feasible routes that maximize the total reward or profit.

$$max \sum_{i=2}^{n-1} \sum_{k=1}^{m} r_i \cdot y_{ik} \tag{1}$$

3 Developed Tool

3.1 The Genetic Algorithm

The Genetic Algorithm (GA) is a search heuristic that imitates the natural process of evolution as it is believed to happen to all the species of living beings. This method uses nature-inspired techniques such as mutation, crossover, inheritance and selection, to generate solutions for optimization problems. The success of a GA depends on the type of problem to which the algorithm is applied and its complexity.

In a GA, the chromosomes or individuals are represented as strings which encode candidate solutions for an optimization problem, that later evolve towards better solutions. Designing a GA requires a genetic representation of the solution domain, as well as a fitness function to evaluate the solutions produced. The GA evolutionary process starts off by initializing a population of solutions (usually randomly), which will evolve and improve during three main steps:

– Selection: a portion of each successive generation is selected, based on their fitness, in order to breed the new, and probably better fit, generation.
– Reproduction: the selected solutions produce the next generation through mutation and/or crossover, propagating the most crucial changes to the future generations by inheritance.
– Termination: once a stopping condition is met, the evolutionary process ends.

3.2 Algorithm Description

The developed genetic algorithm consists on three components. The most elementary one is the chromosome, which represents a set of vehicles and their respective routes. The next component is the evolutionary process, responsible for executing crossovers and mutations within the population of chromosomes. The third and last component of the algorithm controls the evolutionary process, ensures the validation of chromosomes according to the limitations imposed by the TOP and its instances, and also carries out the evaluation of the chromosomes based on a fitness function.

In the GA presented here, a possible solution for the TOP is represented in the form of a chromosome. At each new generation, a renewed population of chromosomes is obtained, each one being a valid solution. A valid solution must contain one route for each vehicle available, and each route includes a sequence of customers to be visited under a given time limit. An example of valid solution is showed in figure 1. There are six customers denoted as numbered vertices and two vehicles are available. There is also a starting point (S) and an ending point (E). In this case, vertex 6 is not included in any route because otherwise it would turn the solution invalid by violating the time constraint.

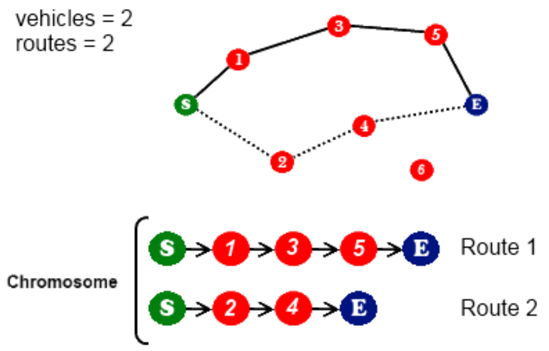

Fig. 1. Representation of an example valid solution for an instance with six vertices and two vehicles available.

While addressing the team orienteering problem, the fitness function of the algorithm was set to correspond to the sum of all the collected rewards in each customer visited during the identified routes.

In order to produce solutions for the TOP, the algorithm starts off by generating an initial population of valid chromosomes. Then, some genetic operators are applied to the population in order to promote their evolution towards better fitness levels. This evolution process is repeated until a stopping criteria is met.

As explained before, the chromosomes in the GA contain routes to be assigned to the available vehicles in a certain TOP instance. Consequently, the creation of a new random chromosome is in fact the creation of a group of random valid routes. All these routes include the required starting and ending points. In order to assemble a route, customers are added in from an availability list that is common to all routes. The attempt to add a customer to a route is only successful if that addition keeps that route feasible while not exceeding the given time budget. Once added to a route, a customer is then removed from the list and marked as checked. Each and every customer in the list is tested for an insertion in the current route. A chromosome must contain as much routes as the number of vehicles available in a chosen TOP instance. Figure 2 presents a simple scheme of how a random route is created. The algorithm uses parallel processing in order to assemble all the routes in a chromosome.

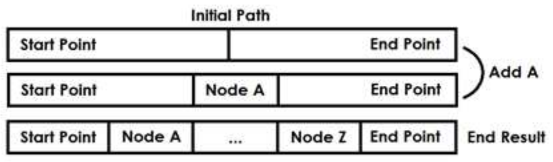

Fig. 2. Creation of a random route.

In respect to the evolutionary process, it includes two genetic operations: crossover and mutation. The crossover procedure is done by exchanging routes between two chromosomes, resulting in the creation of two new chromosomes (see figure 3). The routes to be exchanged are randomly selected, yet entire blocks of consecutive routes are copied to the new chromosomes, as it can be observed in figure 3.

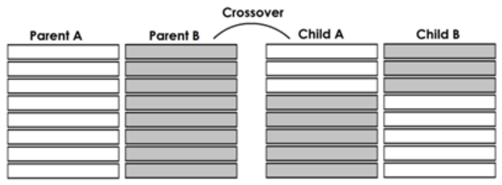

Fig. 3. Crossover procedure between two parents, resulting in two children chromosomes (each bar represents the route of a vehicle).

The chromosomes used for crossover are chosen based on a roulette-wheel selection, also known as fitness proportionate selection, which assesses the probability of a chromosome being used in combinatorial methods. Therefore, a chromosome with high fitness level is more likely to be selected as a parent for the next generation. The number of selected chromosomes must be an even number since these chromosomes are distributed in two lists of parents: Parents A and Parents B. In equation 2, the probability p of a chromosome i to be selected is denoted as p_i, the fitness of that chromosome is f_i and N is the population size.

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{2}$$

The other genetic operation used is mutation and it consists on the removal of a random customer from a randomly chosen route within a chromosome. Then, an attempt is made in order to insert one or more customers from the availability list of the chromosome. The customers in this list are checked one by one in a random order, and when the current customer represents a valid option, it is added to the route. During this process, an attempt is made in order to add as much new customers to the route as possible. In figure 4, a simple mutation is presented, where only one customer is removed from a route. There is also the possibility to perform more complex mutations by removing more than one customer from a route, and the process is executed in a similar way as in the simple mutation.

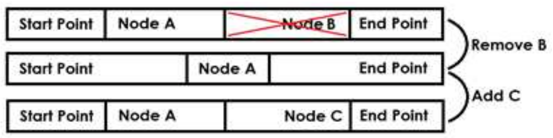

Fig. 4. Scheme of a single-switch mutation.

There are two special classes of chromosomes within the population, the elite and the sub-elite groups, to which different rules are applied. The elite class is the group of the most fit chromosomes within the population in a certain generation, and these chromosomes are immune to mutation until they are replaced by new chromosomes in further generations. As for the sub-elite class, it includes the fittest chromosomes immediately after the elite ones. This group is kept intact during the crossover phase but suffers mutation right after. During the evolutionary process, the resultant chromosomes from both crossover and mutation processes are kept, even if they have less fitness than the chromosomes that originated them.

3.3 Software Developed

The presented algorithm was implemented using the JAVA Swing Framework, and the resulting software tool incorporates a Graphical User Interface that allows adjustment of various parameters. A representation of the software functional process is given in figure 5.

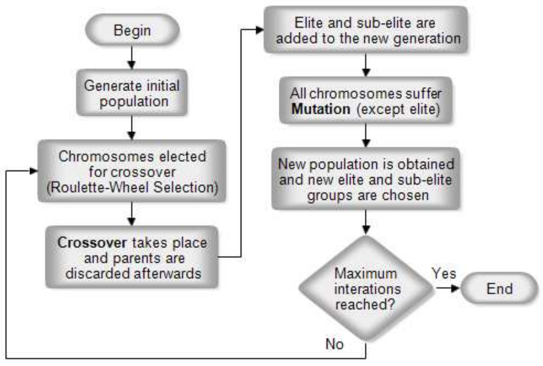
Begin
Generate initial population
Chromosomes elected for crossover (Roulette-Wheel Selection)
Crossover takes place and parents are discarded afterwards
Elite and sub-elite are added to the new generation
All chromosomes suffer Mutation (except elite)
New population is obtained and new elite and sub-elite groups are chosen
Maximum interations reached?
Yes
End
No

Fig. 5. Simplified representation of the software functional process.

4 Computational Test Results and Discussion

In order to assess the performance of the algorithm, a series of computational experiments around the TOP were performed using the developed software tool. These experiments were conducted on 24 of the 320 benchmark TOP instances published by Chao et al. (1996). The instances were chosen in a semi-random way within four different sets, aiming to introduce diversity and different degrees of difficulty while testing the algorithm.

The results achieved during the tests were matched against the ones obtained by the Chao et al. (1996), hereafter referred to as CGW. Comparisons were also carried out with the results presented by Tang and Miller-Hooks (2005), hereafter referred to as TMH, and also the results produced by the algorithms presented by Archetti et al. (2006), hereafter referred to as AHS. The tests were run on a desktop computer with an Intel Pentium Core2Quad Q6600 2.40GHz processor and 4GB of RAM.

During the tests, the maximum number of generations to be produced for each instance was set to be the stopping criterion, with its value limited to 10.000. The number of vertices (customers) in each set of instances includes the starting and ending points. There are 100 vertices in the first set, 66 in the second, 64 in the third, and 102 in the fourth set. An instance is characterized by a number of vehicles, varying between 2 and 4, and by a time limit (Tmax).

In order to execute the tests, it was considered a total population of 220 chromosomes, with 10 being in the Elite and another 10 in the Sub-Elite. Also, per each generational cycle, 100 Crossovers and 210 Mutations occurred.

We ran our algorithm ten times on each selected benchmark instance. The results achieved with our algorithm, hereafter referred to as GATOP-2 [3], in the tested instances, are presented in table 1. The values *fmin* and *fmax* are

[3] GATOP-2 is preceded by GATOP-1 - a previous work in the GATOP project yet to be presented in ICORES 2013 under the title ”Developing tools for the team orienteering problem - A simple genetic algorithm”.

respectively denoted as the minimum and maximum fitness values obtained for each instance. The value *fmin* can be considered a guaranteed value, reflecting the overall performance of the algorithm, along with the average fitness value. The value *fmax* represents the ability of the algorithm to reach good solutions. It is obtained by running the algorithm more than once and taking benefit of the randomness at the expense of a larger computational time.

In table 1, the best scores found for each instance are displayed in bold print, and were mostly set by AHS. As it can be observed, on 14 of the 24 instances, our algorithm equalled the best scores. In particular, our scores were equal to AHS 14 times, and scored the same as TMH and CGW in 7 and 5 instances respectively. In addition, GATOP-2 outperformed TMH for 14 times and CGW for 16 times.

In respect to the best scores, GATOP-2 fell behind on 10 instances, with a maximum error of 56 and an average error of 18.7, considering solely the results on those same ten instances. These error values are inferior to the ones produced by TMH and CGW, while comparing them to the best scores.

An overview of the results produced with GATOP-2 may be sufficient to infer about its performance, particularly in terms of consistency. We confirmed the occurrence of a gap between the *fmax* and *fmin* values in 17 instances, with a maximum value of 125. The average value for the gaps is 36.17. Comparing these values, for example, to the ones outputted by the overall best algorithm presented by AHS, the SLOW VNS_FEASIBLE, GATOP-2 produced much higher gaps between the maximum and minimum scores in the tested instances, since the SLOW VNS_FEASIBLE presents a highest gap value of 18 and an average gap of 6. Therefore, GATOP-2 is less consistent.

During the tests, the maximum computational time of a run with the adopted settings did not exceed 21 minutes, but since the best results were achieved in less than 5.000 iterations, in a practical sense, it can be assume GATOP-2 would achieve the same results in half the time spent on the tests.

We believe the crossover procedure to be the main reason for our algorithm to fail in achieving better results, since the chromosomes that are designated to be parents are discarded immediately after all the crossovers are performed, even if they have a higher fitness than their children. This may lead to a considerable loss of potentially good solutions that would benefit from a later mutation and probably evolve to better solutions.

In respect to the mutation procedure, we opted to decrease the complexity and computational time during the tests, and so we kept the single-switch mutation. By choosing this method, the mutations were accomplished much faster but with lower optimization when compared to a multi-switch mutation, where multiple customers from one or more routes are replaced. Only further tests would confirm if GATOP-2 would produce better results while using more complex mutations.

Another aspect that have an impact on the results is the population size and also the number of elite and sub-elite chromosomes. A series of preparatory tests confirmed that assumption. During those tests, the values of the considered

Table 1. Results achieved with GATOP-2 in the selected benchmark instances.

Instance	GATOP-2			AHS	TMH	CGW
	Avg	*fmin*	*fmax*			
p4.2.e	598.5	570	**618**	**618**	593	580
p4.2.n	1063.5	990	1115	**1171**	1150	1112
p4.3.f	571.8	555	**579**	**579**	**579**	552
p4.3.i	786.6	768	806	**807**	785	798
p4.4.l	841.8	815	879	**880**	875	847
p4.4.q	1110.8	1063	1131	**1161**	1124	1084
p5.2.e	180	180	**180**	**180**	**180**	175
p5.2.m	852.5	810	**860**	**860**	**860**	855
p5.3.h	260	260	**260**	**260**	**260**	255
p5.3.v	1393	1375	1415	**1425**	1410	1400
p5.4.o	682.5	675	**690**	**690**	680	675
p5.4.t	1129	1100	**1160**	**1160**	1100	**1160**

Instance	GATOP-2			AHS	TMH	CGW
	Avg	*fmin*	*fmax*			
p6.2.j	936.6	900	**948**	**948**	936	942
p6.2.n	1228.8	1200	1242	**1260**	1260	1242
p6.3.i	642	642	**642**	**642**	612	642
p6.3.l	985.2	966	**1002**	**1002**	990	972
p6.4.k	528	528	528	528	522	**546**
p6.4.n	1068	1068	**1068**	**1068**	**1068**	**1068**
p7.2.e	290	290	**290**	**290**	**290**	275
p7.2.r	1035.7	974	1077	**1094**	1067	1082
p7.3.g	344	344	**344**	**344**	**344**	338
p7.3.s	1033	1004	1064	**1081**	1061	1064
p7.4.i	365.7	363	**366**	**366**	359	338
p7.4.t	1041.2	1014	1058	**1077**	1067	1066

Table 2. Statistics on the results achieved with GATOP-2 in the tests.

Equal to Best Score	14
Equal to AHS	14
Equal to TMH	7
Equal to CGW	5
Better than TMH	14
Better than CGW	16
Max. Error in respect to Best Score	56
Average Error in respect to Best Score	18.6
Max. gap (*fmax-fmin*)	125
Average gap (*fmax-fmin*)	36.17

parameters were increased from test to test, and GATOP-2 outputted better scores accordingly. Although those values could be raised until the best scores were achieved, that would require a large amount of computational time, turning it into a non-practical method to solve the TOP in a real-world situation. This additional computational time is due to the large amount of crossovers and mutations that occur during each generation. One way to tackle this problem is to determine a better balance between the population size and the quantity of crossovers and mutations. That change would speed up the reproduction process in the algorithm.

5 Conclusions and Future Work

In this paper we presented a genetic algorithm as a solution method for the Team Orienteering Problem (TOP). The achieved results demonstrate that our algorithm is fairly efficient, attaining the best-known solutions in more than half of the tested instances, and scoring near the best in the remaining instances. There are still some enhancements that can be done to improve the results, like modifying the crossover and mutations procedures. Improvements can also be achieved by finding a better balance between parameters such as the total population size and the number of elite and sub-elite chromosomes. A possible way of doing this is to use dynamic parameters to set the behaviour of the evolution process within the genetic algorithm. This could be achieved by implementing a Machine Learning algorithm that would tune the parameters of the genetic algorithm by evaluating its performance during the tests and would apply the best parameter configuration to overcome adversities while aiming for better results.

The assessment of the software tool developed for the presented study was important in order to identify its functionalities, advantages and limitations. Future experimentations will focus on the usage of the C++ programming language, which might perform faster than JAVA.

With these experiments, we were able to improve our knowledge in the TOP, which is certainly a good contribute for the GATOP project. In future work, it is our plan to continue on the development of new strategies to apply in genetic algorithms in order achieve better solutions for the TOP and its variants with more constraints.

Acknowledgements This study was partially supported by the project GATOP - Genetic Algorithms for Team Orienteering Problem (Ref PTDC/EME-GIN/120761/2010), financed by national funds by FCT / MCTES, and co-funded by the European Social Development Fund (FEDER) through the COMPETE Programa Operacional Fatores de Competitividade (POFC) Ref FCOMP-01-0124-FEDER-020609.

References

Archetti, C., Hertz, A., Speranza, M. G.: Metaheuristics for the team orienteering problem. Journal of Heuristics. 13, 49–76 (2006)
Butt, S. E., Cavalier, T. M.: A heuristic for the multiple tour maximum collection problem. Computers and Operations Research. 21, 101–111 (1994)
Chao, I. M., Golden, B., Wasil, E. A.: Theory and Methodology - The Team Orienteering Problem. European Journal of Operational Research. 88, 464–474 (1996)
Ke, L., Archetti, C., Feng, Z.: Ants can solve the team orienteering problem. Computers and Industrial Engineering. 54, 648–665 (2008)
Souffriau, W., Vansteenwegen, P., Vanden Berghe, G., Van Oudheusden, D.: A Path Relinking Approach for the Team Orienteering Problem. Computers & Operations Research, Metaheuristics for Logistics and Vehicle Routing. 37, 1853-1859 (2010)
Tang, H., Miller-Hooks, E.: A tabu search heuristic for the team orienteering problem. Computers and Operations Research. 32, 1379–1407 (2005)
Tasgetiren, M. F.: A Genetic Algorithm with an Adaptive Penalty Function for the Orienteering Problem. Journal of Economic and Social Research. 4 (2), 1–26 (2002)
Vansteewegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D.: A guided local search metaheuristic for the team orienteering problem. European Journal of Operational Research. 196 (1), 118–127 (2009)