A Comparison Between Optimization Tools to Solve Sectorization Problem

Aydin Teymourifar[1,2][0000−0002−0285−0266], Ana Maria Rodrigues[2,4][0000−0003−1070−3626], José Soeiro Ferreira[2,5][0000−0002−7552−9924], and Cristina Lopes[6][0000−0002−4833−470X]

[1] CEGE - Centro de Estudos em Gestão e Economia, Católica Porto Business School
[2] INESC TEC - Institute for Systems and Computer Engineering, Technology and Science, Porto, Portugal
ateymourifar@ucp.pt, aydin.teymourifar@inesctec.pt
[3] CEOS.PP - Center for Organizational and Social Studies of Porto Polytechnic, Porto, Portugal
ana.m.rodrigues@inesctec.pt
[4] FEUP - Faculty of Engineering, University of Porto, Porto, Portugal
jsf@inesctec.pt
[5] ISCAP - Institute for Accounting and Administration of Porto, Porto, Portugal
cristinalopes@iscap.ipp.pt

Abstract. In sectorization problems, a large district is split into small ones, usually meeting certain criteria. In this study, at first, two single-objective integer programming models for sectorization are presented. Models contain sector centers and customers, which are known beforehand. Sectors are established by assigning a subset of customers to each center, regarding objective functions like equilibrium and compactness. Pulp and Pyomo libraries available in Python are utilised to solve related benchmarks. The problems are then solved using a genetic algorithm available in Pymoo, which is a library in Python that contains evolutionary algorithms. Furthermore, the multi-objective versions of the models are solved with NSGA-II and RNSGA-II from Pymoo. A comparison is made among solution approaches. Between solvers, Gurobi performs better, while in the case of setting proper parameters and operators the evolutionary algorithm in Pymoo is better in terms of solution time, particularly for larger benchmarks.

Keywords: Sectorization · Optimization · Pulp · Pyomo · Gurobi · Pymoo

1 Introduction

The main objective of sectorization problems (SPs) is to divide a large territory into smaller sectors according to criteria like equilibrium and compactness [1, 6]. They have many applications in designing territories for airspace [7], commerce [8], electrical power [9], forest [10], healthcare [11, 12], sales [13, 14], school [15], social service [16], politic [17, 18].

Real-life applications of SPs are usually large, and models are generally in the form of linear, quadratic, and non-linear optimization. As in many other fields of operational research, the efficiency of methods and solution times are important matters.

In this study, single-objective (SO) and multi-objective (MO) models are presented for a basic SP, in which, centers are known beforehand. Then different solvers and metaheuristics are used for their solution. The main differences of this study from the previous ones can be summarized as follows: Pulp, Pyomo, intlinprog, Gurobi, and a genetic algorithm (GA) from Pymoo are used to solve these models and the results are compared. MO model is discussed and the non-dominated sorting genetic algorithm (NSGA-II) and the reference point based non-dominated sorting genetic algorithm (RNSGA-II) are used to solve it. It is shown that the RNSGA-II, which has not been used to solve sectorization problems before, has not a good performance as much as NSGA-II.

2 Models Description

This section describes the models of the SP. In a region, there are n points to be assigned to k sectors. The coordinates of points and center of sectors are known beforehand. The related decision variable is defined as follows:

$$x_{ij} = \begin{cases} 1, & \text{if } \textit{point i is assigned to sector j} \\ 0, & \text{otherwise.} \end{cases} \quad i = 1,...,n, j = 1,...,k. \tag{1}$$

Some of the used notations are summarized in Table 1.

Table 1. Used notations

Notation	Description
n	Total number of points
k	Total number of sectors
x_{ij}	Decision variable about assignment of point i to sector j
c_j	Center of sector j
$d_{i_{c_j}}$	Euclidean distance between each two points i and the center of sector j
Q	Target number of points per sector
q_j	Number of points in sector j
$\bar{q}$	Average number of points in sectors
D_j	Total distance of the points in sector j from the center
τ_{equ}	Tolerance for the equilibrium criteria
τ_{com}	Tolerance for the compactness criteria
f_1	Single-objective function related to compactness
f_2	Single-objective function related to equilibrium
f	Multi-objective function related to compactness and equilibrium
ST	solution time

2.1 SC: SO model to minimize compactness

In this model, as defined in Equation 2, compactness is the objective function, while equilibrium is satisfied with a constraint. Compactness is defined based on measuring the total distance of the points to the center of the assigned sector [19, 20].

$$f_1 \; = \; Minimize \; \sum_{j=1}^{k} \sum_{i=1}^{n} d_{ic_j} x_{ij} \tag{2}$$

There are also some constraints, which are defined as in Equations 3, 4, 5 and 6.

Each point should be assigned to only one sector, which is ensured with Constraint 3.

$$\sum_{j=1}^{k} x_{ij} = 1, \quad \forall i = 1, ..., n \tag{3}$$

Constraint 4 is used to assign at least one point to each sector.

$$\sum_{i=1}^{n} x_{ij} \geq 1, \quad \forall j = 1, ..., k \tag{4}$$

With Constraint 5, lower and upper limits are determined for the number of points that can be assigned to each sector, and in this way, the equilibrium is provided within an interval.

$$Q(1 - \tau_{equ}) \leq \sum_{i=1}^{n} x_{ij} \leq Q(1 + \tau_{equ}), \quad \forall j = 1, ..., k \tag{5}$$

where $0 \leq \tau_{equ} \leq 1$ and $Q = \lfloor \frac{n}{k} \rfloor$ is the target number of points per sector. The domain of the decision variable is defined in Constraint 6.

$$x_{ij} \in \{0, 1\}, \quad \forall i = 1, ..., n, \quad j = 1, ..., k. \tag{6}$$

2.2 SE: SO model to maximize equilibrium

In this model, we aim to maximize equilibrium. So, the objective function, as defined in Equation 7, is to minimize the variance of the number of points in each sector. Compactness is provided by Constraint 8.

$$f_2 \; = \; Minimize \; \sum_{j=1}^{k} \frac{(q_j - \bar{q})^2}{k - 1} \tag{7}$$

where $q_j = \sum_{i=1}^{n} x_{ij}$ and $\bar{q} = \sum_{j=1}^{k} \frac{q_j}{k}$.

$$\sum_{i=1}^{n} d_{ic_j} x_{ij} \leq D_j(1-\tau_{com}), \quad \forall j = 1,...,k \tag{8}$$

where $D_j = \sum_{i=1}^{n} d_{ic_j}$, $\forall j = 1,...,k$ and $0 \leq \tau_{com} \leq 1$. Also, Constraints 3, 4, 5, and 6 are valid.

2.3 MCE: MO model to minimize compactness and to maximize equilibrium

In this model, the objective function as defined in Equation 9, considers both compactness and equilibrium, based on the Pareto optimality concept [20].

$$f \; = \; \textit{Minimize} \; (f_1, f_2) \tag{9}$$

Constraints 3, 4, 5, 6, and 8 are valid.

3 Solution Approaches

We use Pulp, Pyomo, intlinprog, Gurobi, and Pymoo to solve the models. Pulp is an open-source package in Python that can solve linear models. Pyomo is an open-source package in Python that can solve linear, quadratic and non-linear models. Intlinprog is a function in the optimization toolbox of MATLAB to solve linear models. Pymoo is an open-source library in Python that contains single-objective evolutionary algorithms (SOEAs) and multi-objective evolutionary algorithms (MOEAs) and many more features related to optimization such as visualization and decision making [21]. From SOEAs, GA and from MOEAs, NSGA-II and RNSGA-II are used.

GA is one of the most famous evolutionary algorithms that starts with an initial population generation, and then selection, crossover, and mutation operators are used to derive generations and this process continues until the termination condition is reached.

In NSGA-II, the populations of parent and offspring are merged and sorted, then using the non-dominated sorting, Pareto fronts are determined. At the next step, the individuals of the new population are selected. The crowing distance is the concept used for selection. This process continues until the condition of termination is reached [22, 23].

The stages of RNSGA-II are similar to NSGA-II, but it uses a modified survival selection, which is frontwise. The front is split since all individuals cannot survive. The selection of individuals on the splitting front is done based on rank, which is calculated as the euclidean distance to each reference point. The rank of the closest solution to a reference point is equal to 1. The ranking of each solution is the best rank assigned to it [24, 25].

All codes are available via the corresponding author's email address.

4 Experimental Results

We use a system with Intel Core i7 processor, 1.8 GHz with 16 GB of RAM. The results obtained for the SO and MO models are presented in Table 2 and 5, respectively. Python 3.5, a trial version of MATLAB R2020b, and Gurobi 9.0.2 with an academic license are used to obtain the results. It is possible to use different solvers in Pyomo. The presented results in Table 2 are for ipopt. Similar results occur with other solvers that can solve quadratic models.

Benchmarks are shown as $n \times m$, where n and m are the numbers of points and sectors respectively. Benchmarks are generated as 200×10, 500×31, 1000×76, and 2000×200. The coordinates of both points and sectors centers are generated according to the Normal(50,10) distribution. In GA, the half uniform crossover and a random mutation with a rate equal to 0.1 are used. Population size and the number of offsprings are equal to 100 and 10, respectively. The number of iterations is 200 but according to the settings, the search ends when the algorithm reaches sufficient convergence in fewer iterations.

For benchmarks 200×10, 500×31, 1000×76, and 2000×200 the value of Q is equal to 20, 16, 13 and 10, respectively. For these benchmarks, feasible results are obtained when τ_{equ} is in intervals (0.2, 1), (0.27, 1), (0.45, 1) and (0.61, 1), while these intervals are (0, 0.89), (0, 0.95), (0, 0.91), and (0, 0.99) for τ_{com}. The values of τ_{equ} are chosen to be 0.2, 0.27, 0.45 and 0.61, when for τ_{com} they are 0.89, 0.95, 0.91 and 0.99. The reason for this choice is to create the tightest ranges. In the tables, the results that are found by the solvers as an optimal solution are made bold.

If the solvers could not find a feasible solution within 15 minutes, the relevant place in the table is indicated by "-". Since there are more parameters, variables, and constraints in real-life SPs, the models and benchmarks in this study can actually be considered as small ones. Therefore, they are expected to be solved in under 15 minutes. In the tables, solution times (ST) are in seconds.

For Model SC, intlinprog can provide the optimal solution for two benchmarks. In terms of achieving optimal results, with used operators, Pymoo does not outperform solvers. It doesn't even find a feasible solution for two benchmarks. However, when the values of population size and the number of offsprings change, feasible solutions are obtained, which are shown in Tables 3 and 4. For Model SC, Gurobi has the best performance in terms of both results and computation time.

In the model SE, which is quadratic, except for benchmark 200×10, within the determined time, Pyomo and Gurobi are not able to find any result. But for this model, the GA in Pymoo achieved results for all benchmarks at a reasonable time.

In both models, CPLEX has a similar performance to Gurobi in terms of both solutions and time, so its results are not given in the table. The version of the used CPLEX is 12.9.0 with an academic license.

Table 2. Comparison between the results for the SO models, obtained using Pulp, intlinprog, Pyomo, Gurobi, and Pymoo

		Model SC			Model SE	
		f_1	ST		f_2	ST
200×10	Pulp	1588.07	< 1			
	intlinprog	1542.71	14	Pyomo	≈ 0	4.19
	Gurobi	1542.71	< 1	Gurobi	≈ 0	2.94
	Pymoo (GA)	3361	8.30	Pymoo (GA)	3	2.47
500×31	Pulp	2555.76	1200			
	intlinprog	2420.33	519.53	Pyomo	-	-
	Gurobi	2434.95	< 1	Gurobi	-	-
	Pymoo (GA)	-	-	Pymoo (GA)	6.56	2.64
1000×76	Pulp	3389.90	1022			
	intlinprog	-	-	Pyomo	-	-
	Gurobi	3123.4	< 1	Gurobi	-	-
	Pymoo (GA)	-	-	Pymoo (GA)	5.31	3.43
2000×200	Pulp	-	-			
	intlinprog	-	-	Pyomo	-	-
	Gurobi	3363.5	2.56	Gurobi	-	-
	Pymoo (GA)	36837	13.02	Pymoo (GA)	6.83	12.11

In metaheuristics, the level of parameters and used operators affect the results. Examples of this are given in Tables 3 and 4, where the population size and number of offsprings are equal to 1000 and 100, respectively. Also, in these tables, int_ux, int_hux and int_sbx, represent the uniform, half uniform and simulated binary crossover, respectively. Details can be found in [25].

The following tables give the results of metaheuristics in Pymoo library, only for benchmarks 200×10, 1000×76, and 2000×200, due to the page limit. As seen, for the used benchmarks, results can be improved when the population size and number of offsprings are defined at higher levels.

Table 3. Comparison between the results for the model SC obtained using different types of operators

	Model SC			
	Crossover type	Mutation rate	f_1	SC
200×10	int_hux	0.1	3172	7.06
	int_hux	0.2	3264	4.74
1000×76	int_ux	0.1	17486	14.24
	int_ux	0.2	17190	19.12
2000×200	int_hux	0.1	35694	41.73
	int_hux	0.2	35176	108.10

Table 4. Comparison between the results for the model SE obtained using different types of operators

	Model SE			
	Crossover type	Mutation rate	f_2	ST
200×10	int_ux	0.1	2.2	8.28
	int_ux	0.2	2	4.55
1000×76	int_hux	0.1	5.65	20.31
	int_hux	0.2	6.13	14.13
2000×200	int_sbx	0.1	6.68	40.18
	int_sbx	0.2	6.73	53.31

The results for the model MCE obtained by NSGA-II and RNSGA-II are presented in Table 5, where NS represents the number of obtained non-dominated solutions. As seen, in two benchmarks NSGA-II achieves more non-dominated so-

lutions. The Pareto fronts figure by NSGA-II and RNSGA-II for 200×10 benchmarks are shown in Figs. 1 and 2.

Table 5. Comparison between the results for the MO models obtained by NSGA-II and RNSGA-II

		Model MCE	
		NS	*ST*
200×10	**NSGA-II**	14	72.94
	RNSGA-II	6	67.88
1000×76	**NSGA-II**	11	17.02
	RNSGA-II	9	31.70
2000×200	**NSGA-II**	4	65.16
	RNSGA-II	4	36.29

5 Conclusion and Future Work

In this study, Pulp, Pyomo, intlinprog, Gurobi, and a GA in Pymoo were used to solve two SO models, one of which is linear and the other quadratic, designed based on compactness and equilibrium objective functions. For the linear model, Gurobi was generally able to obtain a solution for all benchmarks. It also performed well in terms of solution times. In the quadratic model, both Pyomo and Gurobi were unable to achieve an optimal solution within the defined time, for all but small benchmarks.

With the used parameters and operators, although the used GA from Pymoo did not achieve optimal results, it had a good performance in terms of speed, especially when suitable operators and parameters are defined.

Overall, among the solvers, Gurobi performed better. It is obvious that the results and performance of the solvers vary according to the nature of the problem and the used benchmarks. Results of further benchmarks will be presented in future studies.

Gurobi and CPLEX are capable of parallel computing. We have a plan to use this feature in other studies.

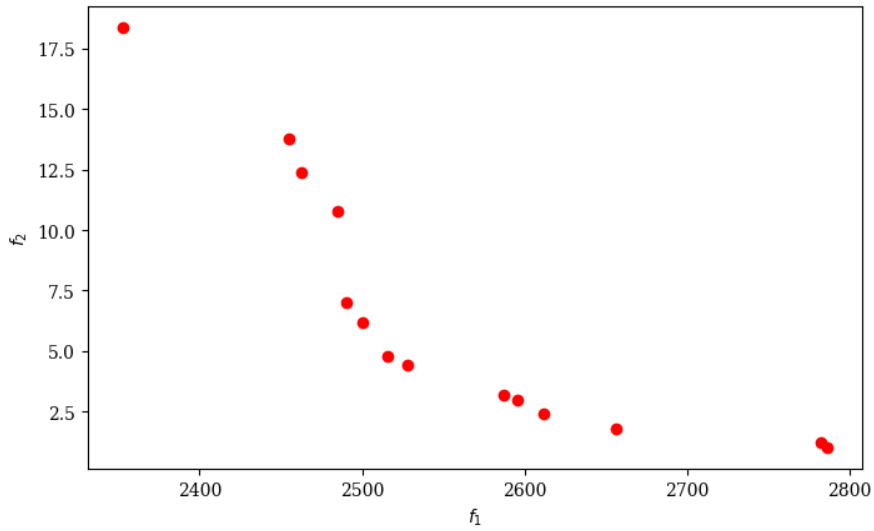

Fig. 1. Pareto front found by NSGA-II for benchmark 200×10

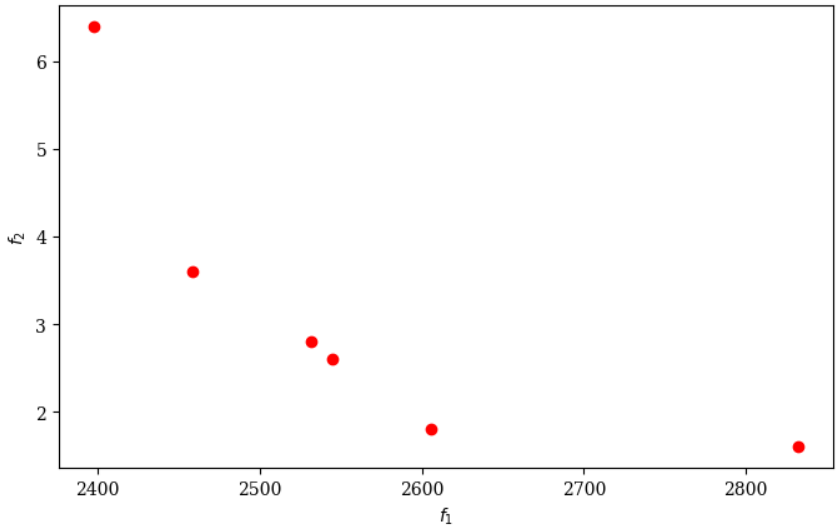

Fig. 2. Pareto front found by RNSGA-II for benchmark 200×10

Also, unlike previous studies, we also considered MO versions of the models and NSGA-II and RNSGA-II, which are available in Pymoo. In two of the benchmarks, NSGA-II performed better than RNSGA-II. The performance of MOEAs can be evaluated based on various indicators. In future studies, different performance indicators will be used.

The authors will consider this study's results for a more effective design of a decision support system dealing with sectorization problems, which they are ending. The results can also provide a reference for researchers who want to use solvers and libraries of this study.

The used parameters, for example, the ones for tolerance intervals, affect the results. In particular, MOEAs have some special parameters that must be defined specifically for the problem. It is planned to make more comprehensive analyzes by considering all parameters and operators used in algorithms.

In this study, only NSGA-II and RNSGA-II are used among MOEAs. Comparison between more algorithms will be made in future studies.

Acknowledgements

Financial support from Fundação para a Ciência e Tecnologia (through project UIDB/00731/2020) is gratefully acknowledged.

This work is financed by the ERDF - European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia within project POCI-01-0145-FEDER-031671.

The authors would like to thank the editor and the anonymous referees for their valuable comments which helped to significantly improve the manuscript. They also express gratitude to Julian Blank for his support about the Pymoo library of Python.

References

1. Teymourifar, A., Rodrigues, A. M., Ferreira, J. S.: A comparison between simultaneous and hierarchical approaches to solve a multi-objective location-routing problem. In Graphs and Combinatorial Optimization: from Theory to Applications, Springer, Cham, 251-263 (2021)
2. Teymourifar, A., Rodrigues, A. M., Ferreira, J. S.: A Comparison between NSGA-II and NSGA-III to Solve Multi-Objective Sectorization Problems based on Statistical Parameter Tuning. In 2020 24th International Conference on Circuits, Systems, Communications and Computers (CSCC), IEEE, 64-74 (2020)
3. Teymourifar, A., Rodrigues, A. M., Ferreira, J. S., Lopes, C., Oliveira, C., Romanciuc, V.: A Two-Stage Method to Solve Location-Routing Problems Based on

Sectorization. In International Conference Innovation in Engineering, Springer, Cham, 148-159 (2021)
4. Teymourifar, A., Rodrigues, A. M., Ferreira, J. S.: A New Model for Location-Allocation Problem Based on Sectorization, Engineering World 3, 92-96 (2021)
5. Rodrigues, A.M., Ferreira, J.S.: Measures in sectorization problems. Operations research and big data, Springer, 203–211 (2015)
6. Rodrigues, A.M., Ferreira, J.S.: Sectors and routes in solid waste collection. In: Operational Research, Springer, 353–375 (2015)
7. Treimuth, T.: Dynamic optimization of airspace sector grouping. Ph.D. thesis (2018)
8. Ríos-Mercado, R.Z., López-Pérez, J.F.: Commercial territory design planning with realignment and disjoint assignment requirements. Omega 41(3), 525–535 (2013)
9. Bergey, P.K., Ragsdale, C.T., Hoskote, M.: A decision support system for the electrical power districting problem. Decision Support Systems 36(1), 1–17 (2003)
10. Shan, Y., Bettinger, P., Cieszewski, C.J., Li, R.T.: Trends in spatial forest planning. Mathematical and Computational Forestry & Natural-Resource Sciences (MCFNS) 1(2), 86–112 (2009)
11. Emiliano, W., Telhada, J., do Sameiro Carvalho, M.: Home health care logistics planning: a review and framework. Procedia Manufacturing 13, 948–955 (2017)
12. Farughi, H., Mostafayi, S., Arkat, J.: Healthcare districting optimization using gray wolf optimizer and ant lion optimizer algorithms (case study: South khorasan healthcare system in iran). Journal of Optimization in Industrial Engineering 12(1), 119–131 (2019)
13. Lei, H., Laporte, G., Liu, Y., Zhang, T.: Dynamic design of sales territories. Computers & Operations Research 56, 84–92 (2015)
14. Ríos-Mercado, R.Z., Fernández, E.: A reactive grasp for a commercial territory design problem with multiple balancing requirements. Computers & Operations Research 36(3), 755–776 (2009)
15. Caro, F., Shirabe, T., Guignard, M., Weintraub, A.: School redistricting: Embedding gis tools with integer programming. Journal of the Operational Research Society 55(8), 836–849 (2004)
16. Lin, M., Chin, K.S., Ma, L., Tsui, K.L.: A comprehensive multi-objective mixed integer nonlinear programming model for an integrated elderly care service districting problem. Annals of Operations Research, 1–31 (2018)
17. Dugošija, D., Savić, A., Maksimović, Z.: A new integer linear programming formulation for the problem of political districting. Annals of Operations Research, 1–17 (2020)
18. Ricca, F., Scozzari, A., Simeone, B.: Political districting: from classical models to recent approaches. Annals of Operations Research 204(1), 271–299 (2013)
19. Kalcsics, J., Nickel, S., Schröder, M.: Towards a unified territorial design approach—applications, algorithms and gis integration. Top 13(1), 1–56 (2005)
20. Romanciuc, V., Lopes, C., Teymourifar, A., Rodrigues, A. M., Ferreira, J. S.,

Oliveira, C., Öztürk, E. G.: An Integer Programming Approach to Sectorization with Compactness and Equilibrium Constraints. In International Conference Innovation in Engineerin, Springer Cham, 185-196 (2021)
21. Blank, J., Deb, K.: pymoo: Multi-objective optimization in python. IEEE Access 8, 89497–89509 (2020)
22. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE transactions on evolutionary computation 6(2), 182–197 (2002)
23. Teymourifar, A., Ozturk, G., Bahadir, O.: A comparison between two modified nsga-ii algorithms for solving the multi-objective flexible job shop scheduling problem. Universal Journal of Applied Mathematics 6(3), 79–93 (2018)
24. Deb, K., Sundar, J.: Reference point based multi-objective optimization using evolutionary algorithms. In: Proceedings of the 8th annual conference on Genetic and evolutionary computation, 635–642 (2006)
25. pymoo: Multi-objective Optimization in Python: http://pymoo.org/ (accessed on July, 2021)