Compacts Typo-Morphologies by Use of Local Search Methods

by Isabelle De Smet, Quentin Meurisse, Vincent Becue, Thomas Brihaye, Jérémy Cenci, David Laplume, Hadrien Mélot, Cédric Rivière
University of Mons (UMONS)

Keywords: compactness, typo-morphologies, housing block, local search.

Abstract: In prospect of sustainable urban densification, a tool aiming to assess and to assist the design of compact housing blocks with a target population density was created and tested in the scope of the CoMod Project. The concept of spatial compactness is here applied, at the architectural scale, on the built environment, the non-built environment and both combined. This approach encourages typo-morphologies which save land and material resources while achieving high energy efficiency. Potential misuse of the concept is prevented by numerous objective criteria notably relative to green areas, projected shadows as well as minimal distances and surfaces to consider.
However, targeting urban compactness faces a difficult conciliation between various quantitative and qualitative parameters. Numerous mathematical tools have already been applied on the problem of urban planning (using optimization methods, multi-criteria decision help, cellular automata, fractal sets, etc.). The study of compact typo-morphologies with the help of graph theory, game theory or local search can help solving the problems that arise when dealing with conflicting criteria. We are currently developing a prototype of software that generates urban blocks using local search algorithms. The generated blocks optimize compactness criteria with respect to a set of comfort, privacy and regulation constraints.

1. Introduction

For decades, architects and urban planners have been interested in density (Clément & Guth, 1995). Densification of territories is the leitmotiv of many urban planners in favour of a *sustainable* development of territories in the face of the peri-urbanization of cities. It is an urban and architectural response envisaged to reduce the increasing use of agricultural and forest land and the sprawl of mobility networks and applicants. Empirical studies (Newman O., 1975; Newman & Hogan, 1981) show that at the scale of a territory, a high density is more favourable for the ecosystem than a low density, which inevitably causes spatial spreading under demographic pressure.

Cities as Assemblages, volume 3
edited by Ilaria Geddes, Nadia Charalambous, Alessandro Camiz
DOI 10.36158/978889295357430
www.tabedizioni.it

How can we urbanise to satisfy the demand for new housing while reducing peri-urbanization and over-consumption of land, energy and materials? How can the architect contribute to this during the design of projects? Acting on the shapes, both volumes and surfaces, could be a solution. Indeed, reducing the built volume and the area of the outdoor spaces, *in judicious way*, helps to limit expenses.

Nevertheless, as shown by Fouchier (Fouchier, 1997), a same building density can lead to very different shapes. For example, the building density is identical for a set of buildings gathered on 10% of the area of a terrain or if they are spread on it. Similarly, a residential tower can have a building population density barely greater than that of a group of residential dwellings, if the surface area of the land on which it is located is high. We deduce that a value of population density cannot be correlated with a typo-morphology. It is therefore difficult to understand the spatial consequences of densification operations from the design stage of a project. The lack of bijective link between typo-morphology and density can prevent efficient assessment of the impact of typo-morphologies in terms of density on the built *and* non-built environment.

Taking that in account, we developed a tool to assist the design of housing blocks following a targeted population density and considering the concept of compactness. Some authors develop this concept on the city scale as a complement to the density (Bertaud & Malpezzi, 2003; Bonin & Tomasoni, 2013; Dantzig & Saaty, 1973; Frankhauser, Tannier, Vuidel, & Houot, 2008; Halleux, 2012; Jenks, Burton, & Williams, 1996) (Katz, 1993; Kirwan, 1992; Maignant, 2005; Pouyanne, 2004). Others apply it on the building scale for energy and material savings (Arantes, Marry, Baverel, & Quenard, 2016; Grenier, 2007; Marique, 2013). The methods for calculating surface compactness are frequently used on the *macro* scale and the ones for calculating the built volume are used in the tool on the *infra-local* scale. More precisely, they are used to study the block based on three components: the plot, the buildings, and also the non-built land created by the interaction between the plot and the buildings. Well-known for its connection with the shapes, the compactness can reduce the land use and the envelope surface of a building through its volume, surface or porosity indicators while exploiting the spatial qualities of compact spaces. Reducing the different values of compactness while satisfying criteria of spatial compactness is one of the objectives of the tool.

Nonetheless, a particular attention is given to the spatial consequences of a drift in the process. Indeed, a set of spatial and morphological constraints must be taken in account during the design of compact housing blocks. We consider the following constraints, in order to avoid an excessive increase of compactness while favouring the spatial qualities of compacts spaces:

- a minimum volume and area of accommodation by type of housing,
- a rate of green spaces area per block, 50% of which consist of a maximum of two spaces of compact shape,
- sufficient light on facades, outdoor and indoor spaces,
- a minimum distance between facades in terms of promiscuity and accessibility for emergency services,
- a feeling of closure of the whole and its components, enabling some porosity between the inside and the outside of the urban block.

Targeting urban compactness faces a difficult conciliation between various quantitative and qualitative parameters. The tool is currently a *static* tool where dealing with all the parameters requires a "trial and error" approach, made tedious by manual encoding. These parameters must be previously computed by the user at each stage of the conception, because of the iterative

nature of the design process. A considerable ease could result from interfacing the tool with a 3D representation software. This would allow for an *automatic* computation of most of the indicators used, based on a CAD model of the housing block, in all its components. It would then become possible to guide the user more effectively as for the interest of a specific modification based on a given configuration of a project. This approach would require a process to handle the indicators and the constraints. So, the tool should include all the properties to implement a true optimization process, in the mathematical sense of the term.

For this purpose, the CoMod Project (Compactness from the angle of mathematic modelling) led by a multi-disciplinary team was created. This project is supervised by three research departments of the University of Mons (Belgium): the Town and Regional Planning Unit (Faculty of Architecture and Urban Planning) and the Algorithms Lab and Effective Mathematics Team (Faculty of Sciences). We are currently developing a prototype of software that generates urban blocks using *local search algorithms*. The generated blocks optimize compactness criteria with respect to a set of comfort, privacy and regulation constraints. Numerous mathematical tools have already been applied on the problem of urban planning (using optimization methods, multi-criteria decision help, cellular automata, fractal sets, etc.). The study of compact typo-morphologies with the help of graph theory, game theory or local search can help solving the problems that arise when dealing with conflicting criteria.

2. Methodology

The initial reflection of software development follows the methodology of building a catalogue of configurations, resulting from a resolutely theoretical experimentation with the previously developed "static" tool (De Smet, 2018). This catalogue, composed of a total of 1674 cases, results from the systematic variation of key parameters on triangular, rectangular and square *fictitious* urban blocks. The area of the considered blocks varies, as well as the depth of the building (from 10 to 18m, by steps of 2m) and the number of levels (from 1 to 8). Limiting the template to 8 levels allows implementation in many urban contexts.

Regarding the layout of buildings, four families of configurations are considered in the catalogue:

- closed (block's perimeter totally built),
- pierced (presence of rifts between the outside and the inside of the block),
- increased (addition of built volume inside the block),
- staggered (presence of various heights of buildings).

These configurations result from following hypothesis:

- a high density combined with a restricted template can be only obtained by use of a larger footprint only for higher constructions;
- the lighting requirements, and thus health requirements, limit the depth of the buildings;
- for a given depth of building, the typo-morphology enabling the biggest footprint favours *de facto* the construction on the perimeter of the urban block as a priority;
- such a construction on the perimeter of the urban block is in line with the common will of the governments to favour the establishments in front of roads (or in alignment with existing buildings);

– a partially closed typo-morphology can lead to determination of private or semi-private outdoor spaces.

This catalogue led to a set of observations related to the evolution of population densities and compactness indicators according to the considered variables, to provide useful guidelines in the context of concrete projects. However, the types of configurations are limited and expanding the catalogue with more parameters would require an unrealistic amount of work.

The dynamic tool in development would make it possible to go beyond the limits of both arbitration and encoding and to develop a real typo-morphological optimization of compact urban blocks. It aims to offer urban blocks' layouts that optimize some criterions and constraints. To achieve this, the tool is using *local search algorithms* (El-Ghazali, 2009; De Beukelaer, Davenport, De Meyer, Fack, 2017). In computer science, *local search* is a method for solving *optimization problems*. That kind of problem can be formulated in this way: given a set of possible *solutions* for a problem, we want to find the solutions that minimize or maximize some criterion called *objective function*. To understand how the local search basically works, we need the notion of *neighbourhood*. Given a solution *s*, a neighbourhood of *s* is a set of solutions that can be reached by performing a small transformation to *s*; an element of this set is called a *neighbour* of *s*. The modification applied to *s* to reach its neighbour is called a *move*. For example, let's imagine that we are on a chessboard's square, on the position of our king. A neighbourhood of this square is the set composed by all the adjacent squares to our king's current position.

Once we have defined the solution set, the objective function and the neighbourhood operation, the basic local search algorithm works as follow: we start from a random solution, we compute its neighbourhood, we evaluate each neighbour with our objective function, the best neighbour becomes our current solution if it is better than our current solution and we iterate this procedure. The algorithm stops if it finds a solution that is better than all its neighbours or satisfy some stop criterion (for example, the search can't take more than a given time).

Besides improvement of the objective function, local search can also try to satisfy some constraints. There are two kinds of constraints: mandatory and penalising constraints. To satisfy a mandatory constraint, local search just discards solutions that don't verify some criterion. Let's return to the example of the chessboard with the objective to find the best next position for our king. An example of mandatory constraint can be "the solution must be a black square". So, the search will ignore the white squares. Penalising constraints give a penalty to the evaluation of a solution that doesn't verify some criterion. Penalising constraints modify the objective function. Local search seeks now to find the best solution that isn't penalised by the constraints.

The main weakness of the basic local search is that it doesn't guarantee that we finally get the best solution. Indeed, the returned solution is better than all its neighbours, not necessarily better than all other solution. It is the best one *locally* and not *globally*. This can be explained by the fact that we don't explore all the possible solutions. To visit more solutions, we need other search methods called *metaheuristics*. El-Ghazali (El-Ghazali, 2009) defines metaheuristics like this: *"Metaheuristic search methods can be defined as upper level general methodologies (templates) that can be used as guiding strategies* [...] *to solve specific optimization problems"*. What we called "basic local search" is a metaheuristic, the Hill Climbing. There exist metaheuristics that accept no-improving neighbours. Accepting no-improving move during the search can lead to better solution than the ones returned by the Hill Climbing.

Let's explain now how we use local search to help us fill an urban block, first only with buildings. We insist on the fact that we use the local search on a specific problem. However, local search is very general and can be used on another problem, with another model or other

constraints. In our context, a solution (an urban block) is modelled as a chequered square or rectangle. Each cell of the quartering can have different colours. A white cell is a no build cell. Other colours/patterns are for buildings. A block of cells with the same pattern represents a building. We currently don't worry about the height of buildings but only about their surface on the urban block. The neighbourhood of a given solution is the set of all the solutions that can be obtained by just changing the colour of a given cell. This action can create a new building in our simulation, expand an existing building, remove a building if we change the cell's colour to white or simply change the colour of a building. Just changing the colour of a building can be interesting because two sufficiently close buildings with the same colour can merge to form a bigger one. We first neglect the compactness of the urban block to focus on its *porosity* (ratio between the built area and the total area of the urban block). Our objective is to minimize this *porosity*.

We don't want to limit ourselves to tile an urban block with squares. We want a model that can work with any kind of tiling, made of regular polygons (hexagons or triangles for example). This allows to test buildings with more varied forms and then other properties on their area and perimeter. For this model, we have implemented a specific object to represent a cell. To create an instance of this object we must specify the number of sides of the cell and the maximum number of adjacent cells. For example, a cell with four sides and maximum eight adjacent cells is a square in a square tiling. A cell with six sides and six adjacent cells is a hexagon in a hexagonal tiling. Each cell is connected to its neighbouring cells. A tiling is a list of cells connected together. Figure 1 illustrates tiling that we have already implemented. To determine the effect of changing the colour of a cell, we need to check its adjacent cells. In addition to the information about its form, a cell object contains a number that symbolises its colour. If this number is equals to zero, the cell is coloured in white and is not built. A cell with a positive number is a build cell and a cell with negative number are the other components of the urban block (-1 for the green spaces for example but we will discuss it in the next section). Let's note that the user can specify the length of cells' sides.

As explained before, local search can work with constraints. In our case, the constraints guide the program to obtain more realistic and interesting solutions. Indeed, with the minimal porosity as only objective, the program can return as a solution an urban block filled with only one big building, or hundreds of tiny buildings without gaps between them. These kinds of solutions don't respect our comfort and privacy constraints, so they must be avoided.

The first mandatory constraint that we have implemented ensures a minimal distance between the facades of different buildings. This constraint takes in input the minimal number of no-build cells that must be present between two build cells. Let's recall that the user can specify the size of cell. As an example, if we work on a square tiling and we want a minimal distance

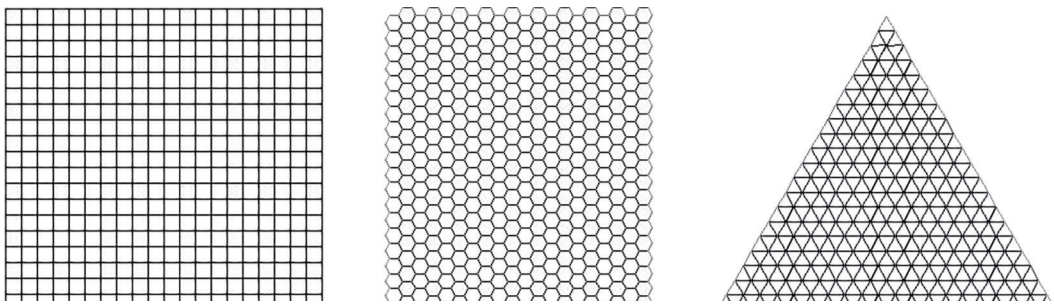

Figure 1.

of four meters, we can work with square of two meters by two meters. So, the constraint must verify that there are two no-built cells between contiguous buildings.

Other constraints are penalising constraints. They penalise solutions if the number of buildings, the total perimeter, the perimeter of each building or the area of each building doesn't belong to a certain range. They take as input the bounds of the range and give as penalty the difference between the observed value and the lower bound if the observed value is lesser than it or the upper bound if the value is greater than it. For example, if we want between six and ten buildings in our urban block, a solution with four buildings will receive a penalty of two. If it has forty-two buildings, it will receive a penalty of thirty-two. Let's just note that the constraint on the perimeter and the area of each building sums the differences between the perimeter or the area of each building and the bounds.

Another feature of our model is the possibility to lock cells. This feature allows us to model the presence of pre-existing buildings or protected green spaces in our urban block. Since those components must be preserved, the neighbourhood operation will not allow changing the colour of a locked cell or expanding a locked building. Locked buildings are coloured with X crosses (see Figures 3 and 4 for examples).

3. Analysis/Results

The implementation methodology of the dynamic tool needs many tests. The observations and analyses of these results guide us in our considerations choices and lead to many possibilities.

Figure 2 illustrates a result obtained by the program on an urban block of 102 meters by 102 meters tiled by squares with sides of 2 meters. The objective is, as explained before, to minimize the porosity of the urban block. Furthermore, we add the following constraints: having between 6 and 10 buildings in our block, the buildings must have a perimeter between 75 and 500 meters and facades must be 4 meters away from each other (or must have two empty squares between them). Each block of cells with the same pattern represents a building.

The constraint on the number of buildings enables to avoid an urban block with numerous small buildings. The constraint on the perimeter enables to avoid results with only one big building and some small ones. As the objective is to minimize the porosity, the tool wants to fill as many cells as possible in the tiling. The results tend thus toward a full urban block. The constraint on the distance between the facades prevents this. As observed on Figure 2, this constraint creates empty spaces that can be interpreted as traffic lanes in our urban block.

Only considering buildings can thus lead to the creation of traffic lanes in our urban block.

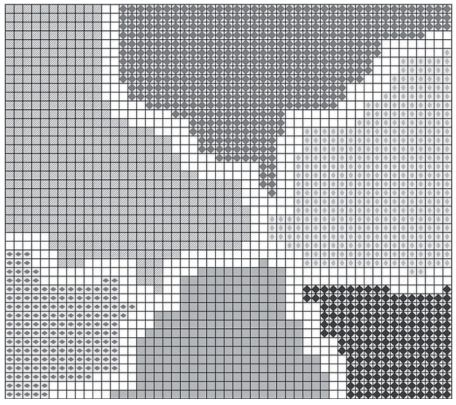

Figure 2.

However, we don't have any control on these traffic lanes and their properties. Moreover, we want a minimal rate of green spaces in our solution and there aren't any places left for them. At this point, we need to introduce other components in the urban block, particularly, the green spaces.

As mentioned before, to distinguish green spaces from buildings, the green spaces are assigned the colour -1, represented by the + cross pattern (see Figures 3 and 4). As mentioned in the introduction, we have constraints on green spaces:

- aim for a certain rate of green spaces on the area of the urban block,
- ensure that maximum two green spaces form 50% of the totality of green spaces.

For more flexibility we have implemented the second constraint in a more general way. The user can specify the rate and how much green spaces must fulfil this rate.

We have currently integrated the green spaces in a quite naive way. During the local search, the program has just a new colour to fill a cell: the colour -1. With this approach, we obtain result as in Figure 3.

This solution is returned with similar constraints as for Figure 2. We require just a number of buildings between 6 and 9 instead of 6 and 10. We impose also 20% of green spaces formed by maximum two green spaces.

We observe in Figure 3 that the tool fills the empty lane formed by the distance between fronts with green spaces. Then, the program creates some spread green to tend forward the targeted 20%. This solution is problematic because it leaves us without the possibility of creating potential traffic lanes. This kind of solution appears because minimizing the porosity and aiming a rate of green spaces is a source of conflicts. On the one hand, we want the biggest possible built area; but on the other hand, we want less buildings to create green spaces. To avoid conflicts, the tool puts green spaces in cells that can't be filled by buildings before anything else. Then, it tries to put green spaces where it can to reach the 20%.

To improve the returned solutions with green spaces, we tried to slightly change the neighbourhood operation used by the local search algorithm. Now, the tool cannot put a green space on an empty cell anymore. They must be "inside" the building. An example of returned solution with the modified neighbourhood is illustrated in Figure 4. Let us remark that, even if we observe empty lanes, the green spaces are still very spread. So, we need to improve our model to get green spaces with more compact forms. An idea is to fill block of cells with green spaces instead of filling it one by one. Another idea is to start from an existing green space and then grow it in a way that preserves its compact form.

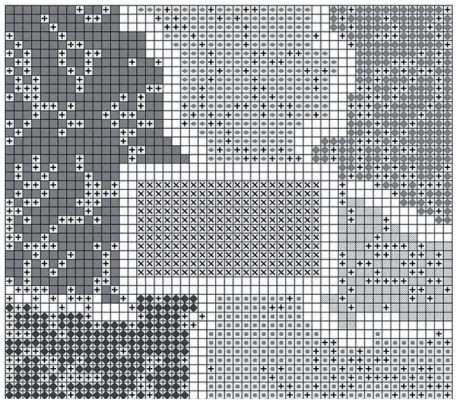

Figure 3.

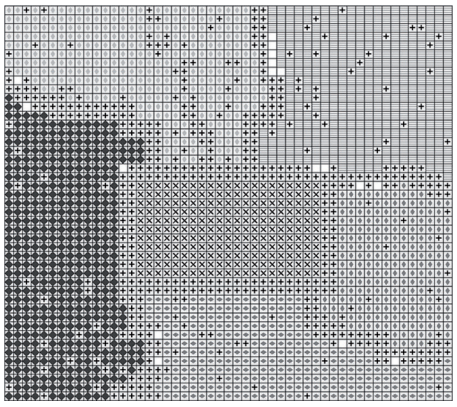

Figure 4.

Moreover, from the firsts tests, the simulations (see Figure 2) show very cut results within the block itself while the outlying bounds of the created buildings always follow the outlying bounds of the block (square here). Enabling the program to return only this kind of solutions risk to standardize shapes outside the block. At this stage of our study, we have decided to neglect this problematic. However, observing the simulations on different scales (on city block scale but also on district and why not on city scale) would allow to solve that problem. The tool could make possible to work on different successive scales. On the district scale, the tool would determine a cut of an area (district) in some urban block with varied forms. On the urban block scale, it would study its content.

4. Discussion/Conclusion

Our tool to assist in the design of compact urban blocks is still in its beginnings. As explained before, it currently aims to minimize the porosity of the urban block with respect to a set of constraints. The obtained results are promising but can be improved, especially the management of green spaces. Although we have neglected the notion of compactness in a first stage, we are aware that compactness is our main objective. The model is frequently enriched with new constraints and objectives to attain the compactness of the urban block.

To achieve compact morphologies, we have to add more constraints on the form of the buildings and the green spaces. Constraint on the perimeter or the area of buildings is a good starting point, but it is not enough. A catalogue of morphologies generated by the tool was sent to the urbanist members of the CoMod project to get remarks and identify the best solutions. These remarks will be used to implement new constraints and refine the model to obtain more realistic results.

Among our future works, we also plan to constraint the depth of building. However, if calculating the depth of a rectangular building is easy, the way to calculate the depth of a more complex polygonal building is not as clear. A first question is thus "how can we formally define the notion of depth for a complex polygonal building?" Our first idea is to define the depth as the diameter of the bigger circumscribed circle in our polygon. To use this definition, we need to find the centre and the radius of this circle, which is not a straightforward problem. In addition to the problem of the definition, comes a problem of computation: "how can we compute *efficiently* the depth?" Indeed, the program is led to know the depth of each building at each step of the search, in order to check if the solution respects the constraints. So, it needs to compute the depth a very high amount of times. This computation may not slow down the search.

As explained before, the program tries to minimize an objective (the porosity of an urban block) and to be penalised the least possible. To achieve this, the program sorts the constraints and the objective. The bigger the penalty, the more "important" the constraint. With this order, a constraint can be totally neglected by the program because its penalty is weaker than another. The program works on this way because it has a very short-term view. Moreover, it is currently difficult to introduce in our model other compactness' criterions of the type "we want that this value is minimal or maximal". We would like to set the constraints and the objective on equal footing and have a multi-objective model that doesn't neglect any criteria or constraints. An idea to do that is to use the mathematical models from the game theory.

References

Arantes L., Marry S., Baverel O., Quenard D. (2016), *Efficacité énergétique et formes urbaines: élaboration d'un outil d'optimisation morpho-énergétique*, in *Cybergeo: European Journal of Geography [En ligne], 777*, p. 29.

Bertaud A., Malpezzi S. (2003), *The Spatial Distribution of Population in 48 World Cities: Implications for Economies in Transition.* Retrieved juillet 3, 2017, from www.alain-bertaud.com: http://alain-bertaud.com/AB_Files/Spatia_%20Distribution_of_Pop_%2050_%20Cities.pdf

Bonin O., Tomasoni L. (2013), *Rendre la ville plus compacte : réflexion autour d'un scénario alternatif à l'augmentation des densités.*

Clément P., Guth S. (1995), *De la densité qui tue à la densité qui paye. La densité urbaine comme règle et médiateur entre politique et projet*, in *Densités et espacements – Les annales de la recherche urbaine,* 67(1), pp. 72-83.

Dantzig G.B., Saaty T.L. (1973), *Compact City: Plan for a Liveable Urban Environnement,* W.H. Freeman, San Francisco.

De Beukelaer H., Davenport G.F., De Meyer G., Fack V. (2017), *JAMES: An object-oriented Java framework for discrete optimization using local search metaheuristics*, in *Software: Practice and Experience,* 47(6), pp. 921-938.

De Smet I. (2018), *Elaboration et expérimentation d'un outil d'évaluation et d'aide à la conception compacts à dominante d'habitat suivant une densité de population cible,* UMONS, Mons.

El-Ghazali T. (2009), *Metaheuristics: from design to implementation,* John Wiley & Sons.

Fouchier V. (1997), *Les densités urbaines et le développement durable – Le cas de l'île-de-France et des villes nouvelles,* Secrétariat général du groupe central des villes nouvelles (SGVN), Paris.

Frankhauser P., Tannier C., Vuidel G., Houot H. (2008), *Une approche multi-échelle de l'accessibilité pour maîtriser l'étalement urbain*, in *International Conference on Mobility and Transport* (p. 18), Technische Universität Münche – Institute for Transportation, München.

Grenier A. (2007), *Ville et énergie. Spécificité et complexité de la question en France*, in Grenier A., Mattei M.F., *La ville dans la transition énergétique – Les annales de la recherche urbaine*, vol. 103, pp. 131-138.

Halleux J.M. (2012, 1-2), *Vers la ville compacte qualitative? Gestion de la périurbanisation et actions publiques*, Belgeo.

Jenks M., Burton E., Williams K. (1996), *The Compact City : A Sustainable Urban Form?,* Spon Press.

Katz P. (1993), *The New Urbanism : Toward an Architecture of Community,* McGraw-Hill Professional, New York.

Kirwan R. (1992), *Urban form, energy and transport: A note on the Newman-Kenworthy thesis*, in *Urban Policy an Research,* 10(1), pp. 6-22.

Maignant G. (2005), *Compacité et forme urbaine, une analyse environnementale dans la perspective d'un développement durable*, in *Développement urbain durable, gestion des ressources et gouvernance*, (p. 17), Lausanne.

Marique A. (2013), *Méthodologie d'Evaluation Energétique des Quartiers Périurbains. Perspectives pour le Renouvellement Périurbain Wallon,* ULiège, Liège, FSA, ArGEnCO-LEMA.

Newman O. (1975), *An ecological model for city structure and development*, in *Ekistics*(239), pp. 258-264.

Newman P., Hogan T. (1981), *A Review of Urban Density Models : Toward a Resolution of the Conflict Between Populace and Planner*, in *Human Ecology,* 9(3), pp. 269-303.

Pouyanne G. (2004), *Des avantages comparatifs de la ville compacte à l'interaction forme urbaine-mobilité. Méthodologie et premiers résultats*, in *«Les cahiers scientifiques du transport»*(45/2004), p. 53.