Particle Swarm for Calibration of Land-Use and Transport Integrated models

Cédric Boittin, Nicolas Gaud, Vincent Hilaire and David Meignan

Abstract

Over the past few years, urban modelers have focused their efforts into making more elaborate simulations, which yield more explanatory power by merging together an increasing amount of urban features. In particular, the integration of Land-Use and Transport Interactions (LUTI) was a key step for urban models. The calibration of these increasingly complex simulations has become a very challenging task, making pure mathematical models inaccurate because of the many assumptions they rely on. This paper aims to extend the tools available to modelers in calibrating LUTI models, in order to manage this problem in a more efficient manner. This paper proposes a metaheuristic approach which considers the calibration problem as a global optimization problem, without relying on assumptions on the model. The chosen optimization algorithm is the Particle Swarm Optimization (PSO). It gradually improves the parameters of the model by testing and comparing the results with survey-based observations.

C. Boittin (Corresponding author) • N. Gaud, V. Hilaire
IRTES-SeT Laboratory, rue Thierry Mieg, 90000 Belfort, FRANCE
Email: cedric.boittin@utbm.fr
N. Gaud
Email: nicolas.gaud@utbm.fr
V. Hilaire
Email : vincent.hilaire@utbm.fr
D. Meignan
Fachbereich Mathematik/Informatik, Universität Osnabrück, Osnabrück, GERMANY
Email: dmeignan@uos.de

1. Introduction

In the current economical context, the management of urban areas is becoming more and more critical. For a more accurate decision making, urban modelers have developed various simulation models which allow testing the impact of potential policies on the city. Over the past few years, modelers have focused their efforts into making more elaborate models, which yield more explanatory power by merging together an increasing amount of urban features. Perhaps the biggest step in this process was the combination of land-use and transportation models into a single model in the late 1950s and onwards (Hansen 1959; Lowry 1964), which mimics the evolution on both sides and, more importantly, the interactions between them. This kind of integrated model is called Land-Use Transport Integrated (LUTI) model. The main objective of a LUTI simulation is to make forecasts of an urban area's evolution in the context of a given scenario, which is a collection of policies to test. This scenario is evaluated through the simulation, allowing to assess the benefits of the considered policies and helping to determine the best option for the decision makers.

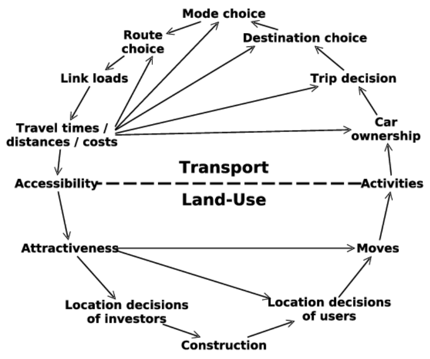


Fig. 1: Land-Use/Transport feedback cycle (from (Wegener and Fürst 1999))

In (Wegener and Fürst 1999), the authors identify the main components and interactions of a LUTI model. These interactions are summarized in figure 1 (drawn from (Wegener and Fürst 1999)). The interface between Land-Use and Transport is clearly defined: the land-use generates localized activities, which impact the way the transportation infrastructures are used. The transportation system simulates the congestion of the network, and the resulting travel times alter the accessibility measures to the features of the city, which will in turn influence how the city actors (households, companies, etc.) make their decisions. In short, the figure shows that these models involve a lot of different aspects, and how complex their interactions are.

Although LUTI models are getting more notoriety, they are still marginal because of their lack of precision, particularly at small scales. This lack of precision makes it hard to draw conclusions about the results of the simulation. In order to increase the accuracy of the simulation, the calibration of the model should be treated very carefully. However, only a few research or practice works have taken any particular attention to the matter. Instead, they focus on each individual submodel and calibrate each of them independently using essentially statistical estimation (Haller et al. 2008; Kakaraparthi and Kockelman 2011). This is a logical extension of the state of the knowledge of the LUTI modelers: they used to run each submodel independently. Nevertheless, when integrating models together, this method comes to its limits.

This paper aims to extend the tools available to modelers in calibrating LUTI models, so to manage this problem in a more efficient manner. In particular, instead of relying solely on model estimations, this paper proposes to use metaheuristical optimization to calibrate the whole LUTI simulation. The advantage with global optimization approaches is that they avoid the definition of an estimation model, which rapidly becomes intractable when the number of interacting submodels increases. The proposed metaheuristic is a Particle Swarm Optimization (PSO) algorithm. This procedure explores the space of possible parameter configurations to find the best one, based on comparisons between the simulation results and observed data of the real system.

The next section details the problem of calibrating LUTI simulations, and presents the global optimization procedure. Then, in Section **Error! Reference source not found.**, the PSO and its application to the calibration problem are explained. Some concluding remarks and perspectives to this work are given in Section **Error! Reference source not found.**.

2. LUTI calibration

2.1 Context

LUTI models are composed of heterogeneous submodels, which interact with each other on many levels. Each submodel has a set of parameters which need to be calibrated. The calibration of LUTI models is hard for several reasons. First, the number of submodels that are involved, and their interactions, makes it difficult to devise a complete and tractable model for the calibration. The model also uses a huge quantity of data that needs to be processed. The simulation run is therefore very time consuming, especially in the transportation affectation step, and it doesn't allow for thousands of trials of parameters values. Moreover, modeling an urban system involves simulating people's behavior, which is impossible to predict exactly. Statistical estimation doesn't apply very well here because of the many assumptions it makes about the nature of the problem and its data, which are in contradiction with the complexity and the highly retroactive effect of each submodel with the others. In this paper, a more general approach is proposed.

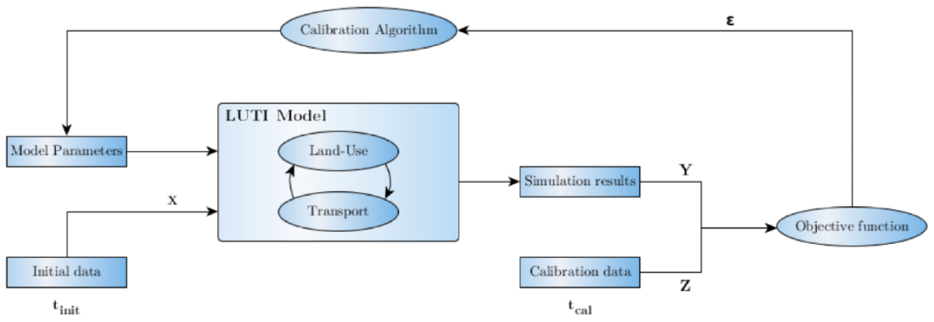
Similarly to Abraham's classification (Abraham 2000) of estimation methods, several approaches to LUTI models calibration can be envisioned. The most simple is the *piece-wise* procedure, where each submodel is calibrated independently. The interactions between submodels are not taken into account, so there is no guarantee that the overall model would even be fit. Another simple and common one is the *limited view* or *black box* approach. This is a purely holistic manner of solving the problem, where all the parameters are tuned at the same time. Both techniques can be combined, resulting in the *sequential* procedure. First a *piece-wise* calibration is used, then a *black box* one is conducted in order to ensure that the overall model is fit. The *simultaneous* approach is the most theoretically appealing one, but also the most difficult to set up. Like in the sequential approach, both the individual submodels and the overall model are taken into account, but they are calibrated at the same time. However, this requires observed data for the intermediary steps of the model, which is not always available.

In simulation in general, it is common to perform calibration using optimization-based techniques, and in particular metaheuristics for optimization (Valente et al. 2008; Fehler et al. 2004). This practice usually gives good results, although it may be criticized, for it is often used in a *black box* manner.

2.2 Process

The proposed calibration process relies on an iterative procedure. The model is expected to be able to reproduce a given state from an earlier known state of the system. Several values of the parameter vector are tried in succession, and the one that gets the model the closest to the known final state is considered the best, and kept. Thus, calibration requires a way to compare the simulation output to the observed situation: the *objective function*. It also requires input data X at date t_{init} that describes the initial state, and observed data Z at any given date $t_{cal} > t_{init}$ to compare with.

The calibration algorithm sets the model parameters, and runs it on the period $[t_{init}, t_{cal}]$. Then, it compares the observed data Z to the simulated data Y from the model's output, and measures an error $\varepsilon = f(Y, Z)$ between them. The goal is eventually to minimize this error to an acceptable



threshold. LUTI calibration then amounts to finding a good objective function and calibration algorithm. This process is summarized in figure 2.

Fig. 2 : LUTI Calibration process

This process can lead to either *black-box* or *simultaneous* calibration. If the comparison between the observed and simulated data produces a single value, then it is a *black-box* calibration. However, if the algorithm allows calibrating multiple objective functions (as in Multi-Objective Optimization), then it is possible to define one objective function for each submodel by comparing different aspects of the output. The resulting procedure would then be *simultaneous*.

This paper's proposal uses a standard PSO implementation, which falls into the *black-box* category: it calibrates all parameters at once without distinction. However, the architecture and agent-oriented implementation favor a *simultaneous* variant, as will be described in Section **Error! Reference source not found.**.

2.3 Software Architecture

No general LUTI tools have reached a consensus in either the research community or among practitioners. Instead, a common approach is to combine existing well-known tools from both the transportation and the land-use simulation fields. In this paper, the authors use the Open Platform for Urban Simulation (OPUS (Waddell et al. 2005)) for the land-use part, and PTV Visum[1], a commercial software, for the transportation part. OPUS is fully written in Python, while Visum can be controlled through a Python API. A first task consisted in the development of a software interface between them.

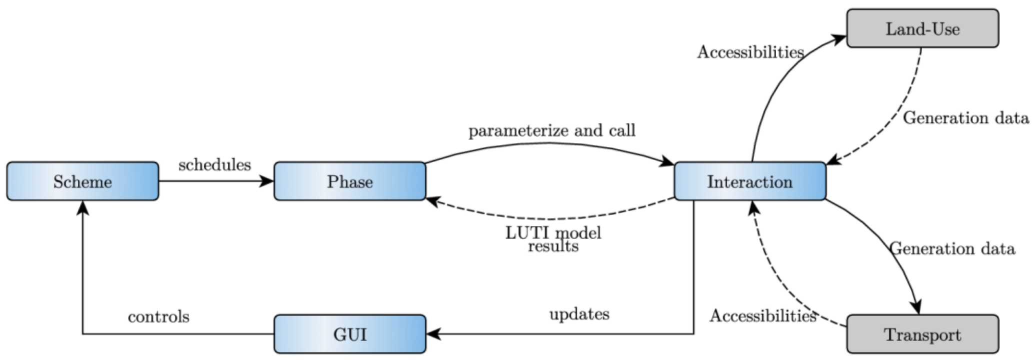
This interface is dedicated to the development of Calibration and/or Validation procedures. It has been designed to assist in two operations:

- The data transfers between the land-use and transport parts: generation data, and accessibility measures, to allow a full LUTI model run in single unified step,
- The design and automated run of a whole Calibration and/or Validation procedure, by scheduling the essential tasks to achieve it.

This interface contains three main components, pictured in figure 3. The first one is dedicated to running the LUTI model. For each software, a manager automatizes the most common operations. These operations are called from a specific object called an *Interaction*, which handles the communication between Land-Use and Transport and eventually defines the workflow of the LUTI model. The *Interaction* can take a set of run parameters as input, which are then used to parameterize the submodels in the context of calibration, for example. It outputs the raw results of the model.

A second layer is added on top of the *Interaction*, for the definition of Calibration and/or Validation procedures. This layer consists in *Phases*, which call the *Interactions* with specific parameters. A *Phase* may for example correspond to the process of calibration or validation. The calibration *Phase* uses the *Interaction* as an *objective function* of the LUTI model, it then only has to set up an optimization algorithm around it, such as the one described in Section **Error! Reference source not found.**. A *Scheme* can schedule the various considered *Phases*, so that a change in the LUTI model design can be quickly and automatically evaluated.

The last component is a graphical user interface which allow the modeler to overview, determine and potentially guide the calibration proce-



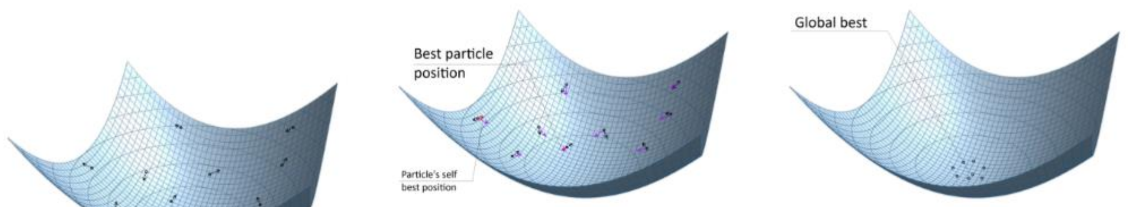
dure.

Fig. 3 : Software Interface

3. Particle Swarm Optimization

3.1 Principles

In this paper, a Particle Swarm Optimization (PSO) (Eberhart and Kennedy 1995; Clerc 2010) algorithm has been developed for the calibration of LUTI models. PSO is a population-based metaheuristic that tries to mimic the behavior of a swarm of individuals (particles) working toward an objective, like foraging bees, for example. A PSO algorithm requires two elements: a space to search in, and a collection of particles. The Search Space is an n-dimensional space containing all possible solutions for the problem: every possible value of the vector of parameters. It is required that each solution can be associated to a single value which indicates its quality. This value is called the *fitness value* and what provides it is called the *objective function*. The particles are objects that move across the search space and communicate with each other. Each particle has a current position x and velocity v, and retains its best solution p found so far: the position which gave the best *fitness* value. At each iteration of the algorithm, the particles are attracted toward a promising neighboring area in the search space, retrieve the fitness value of their current position, and communicate their best known solution g to a few other particles. This way, they move together toward the best solutions, while retaining a wide exploration around these solutions so that it doesn't stick to local optima. Figure **Error! Reference source not found.** pictures the behavior of a PSO on a very simple problem, when each particle communicates its information



(a) Initial state of the PSO: particles have random positions and velocities

(b) After one iteration, particles start to converge toward the most promising direction. A few ones didn't improve their self best and are still attracted toward it.

(c) Final state of the PSO: the particles have progressively converged toward the global optimum and eventually explored the area around it.

immediately to every other one in the

Fig. 4 : Behavior of the PSO solving the sphere function $f(x_1, x_2) = (x_1)^2 + (x_2)^2$, considering immediate transfer of the information

Even though metaheuristics are meant to be general solving methods, their effectiveness might differ depending on those problems' nature. PSO has several advantages: it is efficient with most types of problems, including ones with continuous parameters, multiple local optima, very accidented search spaces, discontinued functions, or a dynamic search space (Clerc 2010). Although being a population-based approach, it doesn't require many individuals to converge (Das et al. 2008), and thus demands less simulation calls than, say, a genetic algorithm. Given the time consumption of one LUTI simulation call, this is especially valuable. PSO has also been extensively used in Multi-Objective Optimization (Lalwani et al. 2013; Liu et al. 2011), which allows for further developments toward *simultaneous* calibration by using one objective function per submodel. PSO can also very easily be distributed over a computer network and thus its computation can be further accelerated.

Parameter-free implementations of PSO exist, however they might require some time at the beginning of algorithm to self-calibrate. Since PSO is used as a calibration algorithm itself, the PSO-related parameters should not influence the final results of the LUTI model very much, so a parametric version is not a problem. Therefore, it is not necessary to spend time calibrating the PSO algorithm itself, instead the parameters are set to fixed a priori values, coming from general best practices.

3.2 Design Choices

As a first try, a standard PSO design (Riccardo et al. 2007) has been implemented. This implementation contains a single swarm of 10 particles. The particles communicate their best known solution at each iteration of the algorithm, and each particle informs 3 other ones on a pattern that is defined prior to the algorithm run. It is ensured that the information from one particle can reach every other one after several iterations.

The particle movements follow the *rectangular* distribution (Clerc 2010), which is the reference implementation for PSO in its random variation. The *rectangular* distribution uses equation (1), x is the position of the particle, v its speed, p the particle's self best and g the particle's known best. The two constants are $c_1 = 0.7$ and $c_{max} = 1.43$ (taken from general indications by (Clerc 2010)):

$$\forall d, \begin{cases} v_{d+1} & \leftarrow c_1 v_d + rand(0, c_{max})(p_d - x_d) + rand(0, c_{max})(g_d - x_d) \\ x_{d+1} & \leftarrow x_d + v_{d+1} \end{cases} \quad (1)$$

When the particle's self best is also the particle's known best, the speed equation becomes equation (2) so that one single solution is not given a too important weight, thus keeping more diversity in the exploration.

$$\forall d, v_{d+1} \leftarrow c_1 v_d + rand(0, c_{max})(p_d - x_d)$$

3.3 Implementation

The PSO algorithm has been developed using SARL (Rodriguez et al. 2014), a holonic agent-oriented language. Agent-oriented programming focuses on implementing the behavior of independent interacting agents, rather than defining and manipulating objects. This kind of approach naturally provides some high-level features such as distribution, interaction, autonomy or dynamic reconfiguration. It focuses on how things happen rather than what can be done. A holon is a conceptual being that is both a whole and a part of something. In terms of agent programming, it allows to define agents composed of other agents and able to act either independently or as the sum of their components.

PSO translates very easily into agent-oriented design: each particle independently moves within the search space, makes decisions about its next move, and communicates with other particles. Moreover, the holonic approach naturally encompasses the multi-swarm approaches (Clerc 2003); Yen and Leong 2009); Peng et al. 2014), since a swarm can be defined as the sum of its particles (or even other sub-swarms). In the context of LUTI modeling, each submodel may be represented by a collection of sub-swarms responsible for its calibration, and the communications between super-swarms ensures the feasibility of the global model. However, this theoretical model remains to be tested.

The PSO hierarchy is pictured in figure . **5**. Each *Model* is an aggregate of *Swarms*, which are aggregates of *Particles*. Thus, the calibration can either focus on one single model or submodels, or treat several submodels at the same time. A *Model* holds a collection of parameters, which determine the search space. The *Particles* have a position in the search space, and this position determines the value of the parameters. Each (sub)model can be calibrated either by a multi- or mono-swarm approach. Communication contexts between *Swarms*, and between *Particles* of a single *Swarm*, are automatically handled by the holonic design. This paper's application consists in a single *Model* and a single *Swarm*, and a *Particle*'s position gives the value for every considered parameter of the model.

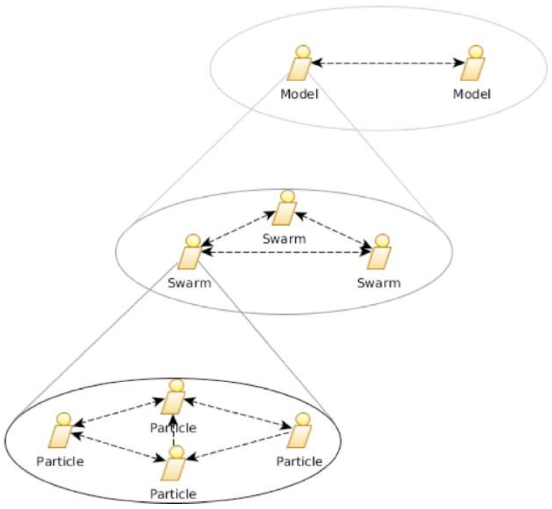


Fig. 5 : Holonic PSO hierarchy. The arrows represent the communication between agents

The SARL software manages a list of simulation calls with the corresponding parameter values. The calibration *Phase* of the LUTI tool then hooks to the SARL software to retrieve these parameter values, uses it to run the *Interaction*, and sends the results back to the PSO.

4. Conclusions

This paper explains the challenges of LUTI Calibration and proposes an alternative to the state-of-the-practice methods for solving it. Indeed, the existing methods have shown their limits. A detailed methodology for the calibration of LUTI models, based on a metaheuristics optimization method, has been given. The Particle Swarm Optimization (PSO) algorithm has been chosen to solve the problem, because it is able to converge within a few iterations, while keeping the broad exploration of a population-based approach. It is also a widely used algorithm, which has proven efficient in solving numerous kinds of problems. A holonic multi-agent PSO has been implemented and described, and it is currently being applied to the calibration of a LUTI model of the Greater Paris.

The method remains to be thoroughly tested on a real case. However, one can build on the simulation literature (Valente et al. 2008; Fehler et al. 2004; Andradóttir 1998) to verify that this kind of approach gives consistent results. PSO comes in many variants, and it is not yet clear to the authors which one would give the best results on LUTI calibration. A dedicated comparison would give valuable insight for choosing one.

Another interesting characteristic about PSO is that it can easily be modified to allow interaction with the user. This has a double advantage: for the algorithm it becomes easier to handle multi-objective functions, and it also has many benefits for the user since it counters the *black-box* effect and allows to manually drive the model toward intuitively interesting directions.

Last, optimization-based approaches can be coupled to statistical estimation techniques, with estimation being valuable for screening the parameters and properly initializing the optimization algorithm.

References

Abraham, J. E. (2000). Parameter Estimation in Urban Models: Theory and Application to a Land Use Transport Interaction Model of the Sacramento, California Region. PhD thesis, University of Calgary, Canada.

Andradóttir, S. (1998). A review of simulation optimization techniques. In Proceedings of the 30th conference on Winter simulation, p151–158. IEEE Computer Society Press.

Clerc, M. (2003). TRIBES, a parameter free particle swarm optimizer. p10–02.

Clerc, M. (2010). Particle swarm optimization, vol 93. John Wiley & Sons.

Das, S., Abraham, A., and Konar, A. (2008). Advances of Computational Intelligence in Industrial Systems. Studies in Computational Intelligence, p1–38. Springer Berlin Heidelberg.

Eberhart, R. C. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In Proceedings of the sixth international symposium on micro machine and human science, vol 1, p39–43. New York, NY.

Fehler, M., Klugl, F., and Puppe, F. (2004). Techniques for analysis and calibration of multi-agent simulations. In Proceedings of the 2004 ESAW, p131–136.

Haller, R., Emberger, G., and Mayerthaler, A. (2008). A system dynamics approach to model land-use/transport interactions on the national level.

Hansen, W. G. (1959). How accessibility shapes land use. Journal of the American Institute of Planners, 25(2):73–76.

Kakaraparthi, S. K. and Kockelman, K. M. (2011). Application of UrbanSim to the Austin, Texas, Region: Integrated-Model Forecasts for the Year 2030. Journal of Urban Planning & Development, 137(3):238–247.

Lalwani, S., Singhal, S., Kumar, R., and Gupta, N. (2013). A comprehensive survey: Applications of multi-objective particle swarm optimization (MOPSO) algorithm. Transactions on Combinatorics, 2(1):39–101.

Liu, Y., Ling, X., Shi, Z., Lv, M., Fang, J., and Zhang, L. (2011). A survey on Particle Swarm Optimization algorithms for multimodal function optimization. Journal of Software, 6(12):2449–2455.

Lowry, I. S. (1964). A model of metropolis. Rand Corporation, Santa Monica, CA, USA.

Peng, M.-Q., Gong, Y.-J., Li, J.-J., and Lin, Y.-B. (2014). Multi-swarm particle swarm optimization with multiple learning strategies. In Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion, p15–16. ACM.

Riccardo, P., Kennedy, J., and Blackwell, T. (2007). Particle swarm optimization. Swarm intelligence, 1(1):33–57.

Rodriguez, S., Gaud, N., and Galland, S. (2014). SARL: a general-purpose agent-oriented programming language. In The 2014 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, Warsaw, Poland. IEEE Computer Society Press.

Valente, P., Pereira, A., and Reis, L. P. (2008). Calibration agent for ecological simulations: a metaheuristic approach.

Waddell, P., Ševcıková, H., Socha, D., Miller, E., and Nagel, K. (2005). Opus: An open platform for urban simulation. In Computers in urban planning and urban management conference, London.

Wegener, M. and Fürst, F. (1999). Land-use transport interaction: state of the art.

Yen, G. G. and Leong, W. F. (2009). Dynamic multiple swarms in multiobjective particle swarm optimization. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 39(4):890–911.