CPlan

An Open Source Library for Computational Analysis and Synthesis

Reinhard Koenig[1]
[1]*ETH Zurich*
[1]*http://www.ia.arch.ethz.ch/koenig/*
[1]*reinhard.koenig@arch.ethz.ch*

Some caad packages offer additional support for the optimization of spatial configurations, but the possibilities for applying optimization are usually limited either by the complexity of the data model or by the constraints of the underlying caad system. Since we missed a system that allows to experiment with optimization techniques for the synthesis of spatial configurations, we developed a collection of methods over the past years. This collection is now combined in the presented open source library for computational planning synthesis, called CPlan. The aim of the library is to provide an easy to use programming framework with a flat learning curve for people with basic programming knowledge. It offers an extensible structure that allows to add new customized parts for various purposes. In this paper the existing functionality of the CPlan library is described.

Keywords: *Evolutionary algorithms, Open source, Planning synthesis, C-Sharp, Spatial configurations*

INTRODUCTION

To be independent from existing CAAD packages we implemented basic geometry objects with a computational geometry library as well as a geometry viewer with corresponding mouse interaction (see section "Geometry library, viewer and mouse interaction"). For evaluating spatial configurations we provide a collection of computational analysis methods (see section "Computational analysis"). For the automatic generation of spatial configurations there are various methods described in section "Generative methods". The synthesis methods of the CPlan library combines everything by using single- or multi-criteria optimization methods (see section "Synthesis methods"). Section "Visualization" describes our approach for mapping a multi-dimensional solution space to 2D by using Self-Organizing-Maps. All parts of the CPlan library are open source and are designed that they can be extended using the provided interface classes.

GEOMETRY LIBRARY, VIEWER AND MOUSE INTERACTION

The basic component is a viewer for 2D and 3D, which includes the usual basic view controls like pan, zoom, and rotate. In addition there is a set of geometry objects that can be visualized by the viewer in a way that the program code to add geometries is minimized and as simple as possible. Basic mouse interaction methods like pick, move, and rotate are already implemented for each geometry object. There are interfaces for lines, surfaces, and bodies that allow a user to add own geometrical objects that can be visualized and used for mouse interaction in the same way as the existing objects. On the project website there are already samples for the introduction how to use the viewer and the geometry classes with integrated mouse interaction. It is basically a very rudimentary cad system (see figure 1).

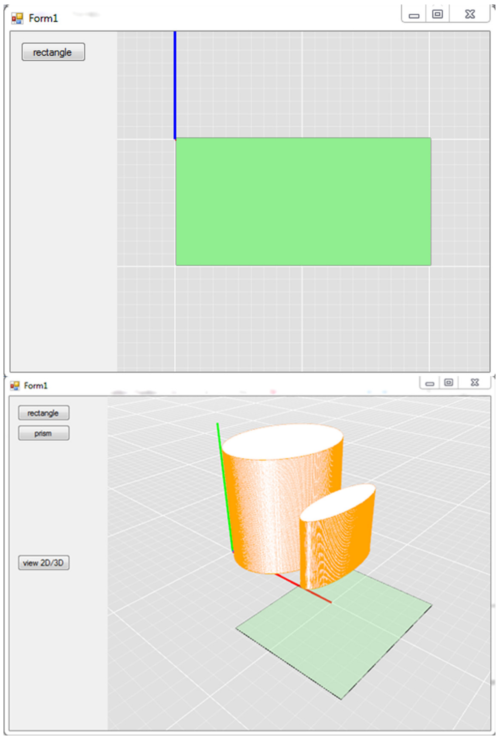

Figure 1
Two examples for using the geometry objects and the viewer. Top: A simple 2D rectangle. Bottom: Objects in 3D view.

COMPUTATIONAL ANALYSIS

A more advanced part of the CPlan library covers various evaluation methods for spatial configurations. So far there are graph measure to calculate centrality measures for street networks as known from Space Syntax (Hillier, 2007; Turner, 2001): Integration (centrality) and Choice (in-betweenness centrality), which can be calculated for custom radii (figure 2b). We also offer the following Isovists field calculations (Batty, 2001; Benedikt, 1979; Turner, Doxa, O'Sullivan, & Penn, 2001) for the analysis of the view field properties: Area, compactness, occlusivity, perimeter, min radial, max radial (figure 3b and c). Furthermore there are visibility graph analysis (Turner et al., 2001) for the evaluation of visual centralities, and solar analysis to calculate the shadowing or the solar exposure during a defined period of time using HDRI maps [3] (figure 3d). All evaluation methods are prepared to be run on a distributed system (LUCI) that is developed for parallel evaluation of larger sets of spatial configurations. This is necessary for the usage of the evaluation methods for objective functions in the context of optimization. The LUCI middleware itself is not part of the CPlan library but a separate project [4].

GENERATIVE METHODS

There are generators for street networks, parceling, building layouts, and building floorplans. Figure 2a shows a street network generated with random parameters using the algorithm from Koenig, Treyer, and Schmitt (2013). Figure 2b illustrates the centrality analysis (choice) of the network. The blocks, which are automatically generated for the street network (figure 2a) can be used for the parceling algorithm from Knecht and Koenig (2012). Based on the block or parcel structure we can generate layouts for building volumes (figure 3a). Figure 3b and c show Isovist analysis and figure 3c a solar analysis for the generated building layout.

SYNTHESIS METHODS

The classes for evolutionary multi-criteria optimization (EMO) are most important ones that enable planning synthesis. They are also the most sophisticated part of the CPlan library. We have not implemented new EMO algorithms, but used a set of well tested existing ones. As blueprint for the classes for populations, chromosomes, and fitness functions we build on the AForge.Net framework [2]. The classes from this framework are adapted for multi-criteria optimization. As multi-criteria selection mechanism we implemented a wrapper to the PISA selectors [5]. This basic setup is applicable for any kind of optimization problem.

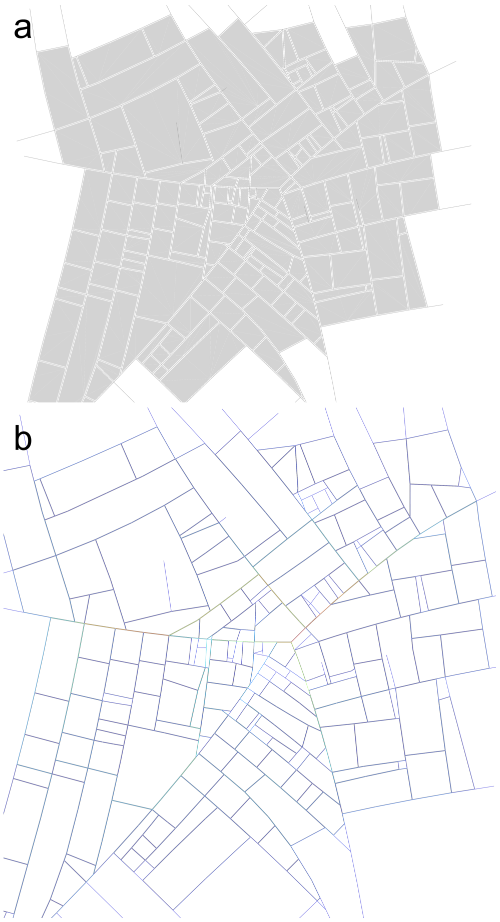

Figure 2
a) Generated street network with blocks, b) Network analysis showing the centrality measure choice.

Beside the evaluation algorithms we added customized algorithms for the generation of street networks and building layouts. These generators are either directly encapsulated in a chromosome (as for the building layouts) or they use the chromosomes as instructions for a more complex embryogeny, which is the mapping process from a genotype to a phenotype. Both algorithms may serve as examples for the implementation of own customized generators for spatial configurations. The provided interfaces for chromosomes and fitness functions ensure that own algorithms are compatible with the population and the selectors, which can remain always the same.

The last two figure show exemplary applications of the CPlan library for planning synthesis in two different urban environment. Existing street networks and building geometries can be imported via dxf. At the top figure only building layouts are synthesized. The figure below shows a process where first the street network is synthesized and second the building layout for the blocks is created.

VISUALIZATION

Another important aspect for dealing with EMO is the possibility to visualize the solution set in a meaningful way. Since there is usually not one best solution, but a set of pareto-optimal solutions, we need a method to show them on a 2D map. Therefore we use SOMs. They are able to reduce a multi-dimensional space to a 2D space, on which the solutions are shown in an understandable way for an urban planner or an architect. To produce SOM visualizations we have coupled the CPlan library to the Databionic ESUM tools (Ultsch & Moerchen, 2005).

WOW TO USE IT

In following we introduce the usage of the CPlan library for design synthesis applications. In the framework of this paper we can only explain the basic concepts. We are continuously working to extend the documentation of the CPlan library and examples for its usage on the project website [1].

The main part for implementing a new optimiza-

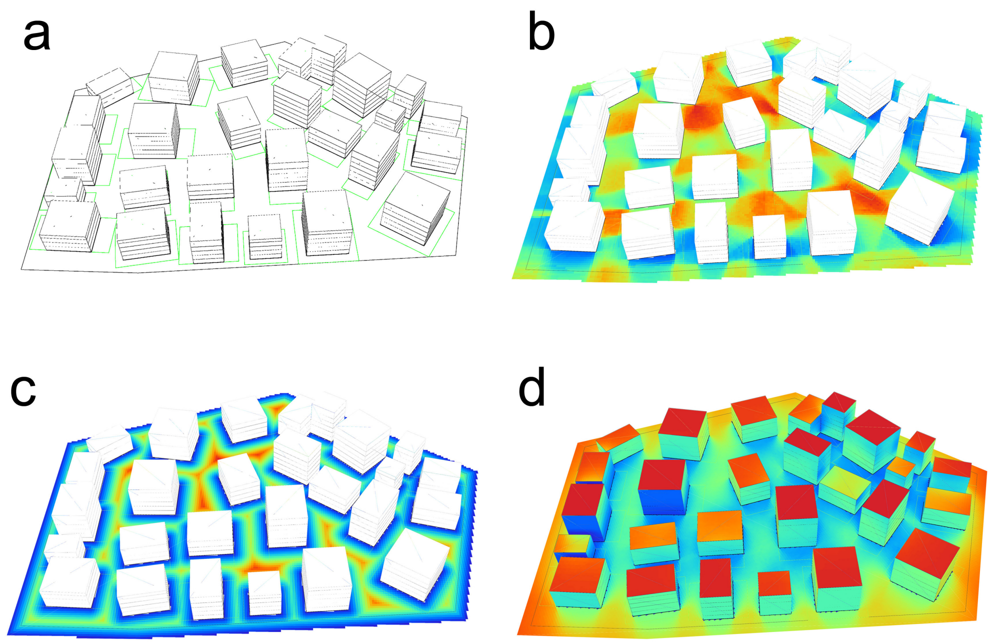

Figure 3
a) Generated building layout, b) Isovist analysis showing the area property, c) Isovist analysis showing the min radial property, d) Solar analysis.

tion problem is to create a custom chromosome, which contains all necessary information for a spatial configuration (e.g. for a street network, a building volume layout, or a floorplan). A spatial configuration is composed out of spatial entities that are stored as gens inside a chromosome. Gens usually represent the smallest spatial entities like street segments, parcels, buildings, or rooms. Gens are manipulated by mutation, crossover, and adaptation operators on the chromosome level. All chromosomes are stored into a population class, from where all operators are called per iteration (also called generation or time step).

The quality of a chromosome, respectively of a variation of a spatial configuration is evaluated by a corresponding fitness function. This function can compute one (or for multi-criteria optimization problems many) objective values, which shall be minimized or maximized during the optimization process. The computational analysis methods of the CPlan library are made for being used with fitness functions.

The structure of the optimization classes can be summarized as follows: A set of chromosomes which represents design variants are stored as a list in a population. The chromosomes itself consist of a list of gens. Each gen stores the information (e.g. parameters) of a spatial entity (e.g. a building). In the following we give some code examples.

To create a new chromosome we inherit from one of the base chromosomes provided by the CPlan library:

```
public class ChromosomeBuildingLayout
 ↪ : ChromosomeBase
```

In the constructor of the custom chromosome usually the restrictions for a spatial configuration are defined:

```
myBuildingLayout = new
 ↪ ChromosomeBuildingLayout (
 ↪ borderPolygon, MinNrBuildings,
 ↪ MaxNrBuildings, Density,
 ↪ MinBuildingSideRatio);
```

A custom fitness function need to inherit from the corresponding interface:

```
public class
 ↪ FitnessFunctionBuldingLayout :
 ↪ IFitnessFunction
```

We create an instance of our fitness function:

```
FitnessFunctionBuldingLayout
 ↪ myFitnessFunction = new
 ↪ FitnessFunctionBuldingLayout();
```

And send the chromosome and the fitness function to the constructor of our population:

```
myPopulation = new
 ↪ BuildingLayoutPopulation(
 ↪ _populationSize, alpha, lambda, mu
 ↪ , curChromosome, fitnessFunction);
```

The population class is always the same for all possible chromosomes. A generation of iteration of the population is executed by:

```
myPopulation.RunEpoch();
```

Whereas the RunEpoch method runs all operators on the chromosomes:

```
public RunEpoch( )
{
    Crossover();
    Mutate();
    Adapt();
}
```

Like the other operators, the Adapt() method can be implemented individually for customized chromosome class. We use it to ensure that a variation is valid, e.g. by a collision detection algorithm, which avoids overlapping of buildings moved randomly by the mutation operator. This makes the optimization more efficient, since it do not need to test all impossible designs.

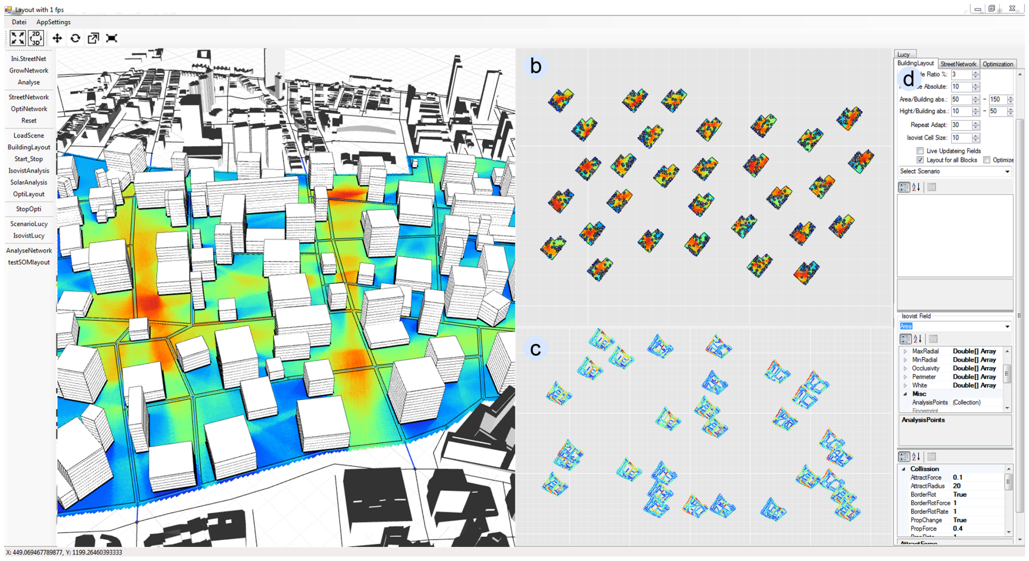

Figure 4
Software prototype showing the main areas of the synthesis systems user interface: a) a 3D view combines one solution out of each archive, design solutions of the archives for b) buildings layouts and c) street networks, and d) fields for the user input of size of population, number of generations, etc.

Figure 4b shows a population of building layouts after a certain number of iterations. Whereas, the precise description in the context of a multi-criteria optimization is, that the set of solutions in figure 4b shows the archive of all best so far found layouts. Figure 3b, c, and d visualize results of possible fitness functions. Theoretically each of the color values in the figures can be used as objective value. This means that we can for example define a fitness function, which shall maximize the area value in the middle of the area of the building layouts in figure 3. As shown by Schneider and Koenig (2012), the results would be, that the buildings are placed at the border of the area after some iterations.

CONCLUSION

The CPlan open source library is an easy to use and well extensible programming framework for computational analysis and planning synthesis. It is published under the GNU Lesser General Public License and available via the project website [1]. The library is written primarily in C-Sharp, while special parts are implemented in managed C++.

There is an active development of the library especially for various application scenarios. The documentation is primarily done by xlm documentation comments directly in the source code. Samples on the project website help for the first steps in programming an own application. We are looking for active developers who support the project or use the framework for aapplication scenarios.

ACKNOWLEDGEMENTS

The authow want to thank the following contributors of the CPlan library: Sven Schneider, Christian Tonn, Martin Bielik, Artem Chirkin, Christine Meixner, Dr. Bernhard Klein, Lukas Treyer, Zeng Wei, Pavol Bielik, Matthias Standfest. The complete CPlan project with all its libraries can be downloaded at www.cplan-group.net.

REFERENCES

Batty, M 2001, 'Exploring Isovist Fields: Space and Shape in Architectural and Urban Morphology', *Environment and Planning B Planning and Design*, 28(1), pp. 123-150

Benedikt, ML 1979, 'To take hold of space: isovists and isovist fields', *Environment and Planning B*, 6(1), pp. 47 - 65

Hillier, B 2007, *Space is the machine: A configurational theory of architecture*, Space Syntax

Knecht, K and Koenig, R 2012, 'Automatische Grundstuecksumlegung mithilfe von Unterteilungsalgorithmen und typenbasierte Generierung von Stadtstrukturen', *Arbeitspapiere Informatik in der Architektur*, 15(15), pp. 1-21

Koenig, R and Knecht, K 2014, 'Comparing two evolutionary algorithm based methods for layout generation: Dense packing versus subdivision', *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 28(03), pp. 285-299

Koenig, R, Treyer, L and Schmitt, G 2013 'Graphical smalltalk with my optimization system for urban planning tasks', *Proceedings of the 31st eCAADe Conference – Volume 2*, Delft, Netherlands, pp. 195-203

Schneider, S, Fischer, JR and Koenig, R 2011, 'Rethinking Automated Layout Design: Developing a Creative Evolutionary Design Method for the Layout Problems in Architecture and Urban Design.', in Gero, JS (eds) 2011, *Design Computing and Cognition '10*, Springer Netherlands, Stuttgart, pp. 367-386

Schneider, S and Koenig, R 2012 'Exploring the Generative Potential of Isovist Fields: The Evolutionary Generation of Urban Layouts based on Isovist Field Properties', *30th International Conference on Education and research in Computer Aided Architectural Design in Europe*, Prague, pp. 355-363

Turner, A 2001 'Angular Analysis', *3rd International Space Syntax Symposium*, Atlanta

Turner, A, Doxa, M, O'Sullivan, D and Penn, A 2001, 'From isovists to visibility graphs: a methodology for the analysis of architectural space', *Environment and Planning B: Planning and Design*, 28(1), pp. 103-121

Ultsch, A and Moerchen, F 2005, 'ESOM-Maps: tools for clustering, visualization, and classification with Emergent SOM', *Dept. of Mathematics and Computer Science, University of Marburg*, pp. 1-7

[1] http://cplan-group.net

[2] http://www.aforgenet.com/framework

[3] http://freac-x.de/freac/drupal/?q=download/hdri_sky_generator

[4] https://bitbucket.org/treyerl/lucy

[5] http://www.tik.ee.ethz.ch/sop/pisa/