EVOLUTIONARY OPTIMIZATION PROCESSES AS DESIGN TOOLS: IMPLEMENTATION OF A REVOLUTIONARY SWARM APPROACH

Author name-surname	Author Company/Institution	Author Email (optional)
Judyta Cichocka	Wroclaw University of Technology	judyta.cichocka@gmail.com
Will Browne	Victoria University of Wellington	will.browne@ecs.vuw.ac.nz
Edgar Rodriguez	Victoria University of Wellington	Edgar.Rodriguez-Ramirez@vuw.ac.nz

Fig 1: Optimization processes in Particle Swarm Optimization: particles and their current velocities.

WHICH ARE YOUR ARCHITECTURAL (R)SOLUTIONS TO THE SOCIAL, ENVIRONMENTAL AND ECONOMIC CHALLENGES OF TODAY?

Research summary

Building sustainable and resilient lives in harmony with the ecosystems and local resources requires a bottom up approach as it starts from the analysis of people needs and social-economic trends. Most current social, environmental and economic challenges have multiple features, such as changes in people lifestyles, urban growth, energy and water expenditures, affordability and quality of living conditions. In order to rationalize these features, complex problem analysis and optimization tools are introduced into the design process. Evolutionary Computation (EC) techniques are considered to be suitable in solving most design problems (Kicinger et al., 2005). However EC solvers are slow (Rutten, 2010), unintuitive, hard to visualize and not verified (Vierlinger, 2013), therefore they are not widely adopted, resulting in many valuable design opportunities being missed. This paper visualizes how biological optimization techniques can assist with architectural design problems. Furthermore, a Swarm Intelligence (SI) approach is introduced as it is hypothesised to be fast, easy to tune and intuitive in its operation. The most popular platform for parametric modelling - Grasshopper® plug-in for Rhinoceros 3D® (Martyn, 2009) was selected as the demonstrator. Experimental results on four domains of increasing complexity show the effects of diversification in the mapping processes of solvers. The introduced Particle Swarm Optimisation (PSO) technique, which relies on SI rather than EC, demonstrated its computational speed, intuitiveness and robustness in complex optimization problems. Such problems increasingly occur in modern architectural practice, so PSO has the potential to become a revolutionary design tool.

Keywords: evolutionary computation, optimization, parametric design, Particle Swarm Optimization

1. Introduction

Architecture is in a continuous evolution. How to enable it to evolve in the era of digital revolution? Novel tools are being created to assist architects in the creation of sustainable and adaptive projects fitting to modern lifestyles and satisfying expanding people needs. New optimization tools are accommodating in aiding and solving architectural design problems, such as "daylight availability, circulation, optimal floor plan layouts, air circulation, adaptation of projects to law restrictions, cost-effectiveness, energy consumption […] and many others" (Cichocka & Musikhina, 2014: 111). Existing optimization techniques that exploit Evolutionary Computation (EC) can address a wide variety of such complex design problems (Kicinger et al., 2005). Nevertheless, EC-based tools are slow (Rutten, 2010) and not validated (Vierlinger, 2013), therefore disengaging and hard to understand. This prevents widespread adoption, thus missing many revolutionary opportunities in the design space. The first objective is to verify and visualise EC-based problem solving methods to understand modern optimisation techniques in the design context and gain insight into their operation. Then the second objective is to improve the speed of optimization processes by introduction of a new technique - Swarm Intelligence (SI). Instead of Darwinian principles of the existing techniques, SI approach is hypothesised to be inexpensive computationally (Kennedy & Eberhart, 1995), easy to tune and intuitive. The most popular application for parametric modelling - Grasshopper for Rhino 3D (Martyn, 2009), was selected as a demonstrator. In this modelling environment the SI approach is implemented on the canvas of provided evaluation tests of the existing alternatives.

2. Research objectives

2.1 Visualization of mapping processes

Currently, in Grasshopper there are three solvers that utilize evolutionary optimization techniques: Galapagos, Goat and Octopus. They are able to address most of the design problems that may be encountered by planners and architects. Although in-depth understanding is not obligatory to use solvers, the potential users are responsible for defining the *Solution Space, Fitness Function* and *Fitness Landscape* (Rutten, 2014). An "aware" user can provide the correct formulation of the above so that the performance of solvers and the quality of results may be significantly increased. However, the lack of validated tests on solvers (Vierlinger, 2013), their unintuitive operation mechanism and numerous tuning parameters, lead to misinterpretations and misuse of EC techniques (e.g. application in brute force search). The first objective of the research is to provide a reliable assessment of the evolutionary solvers in Grasshopper and to assess the black box in their computation in order to prevent practitioners from misuse of EC techniques.

2.2 Implementation of PSO in Grasshopper

The prevalence of EC techniques in optimization tools for parametric modelling is substantiated by their robustness across variety of design problems and by their ability to handle high-dimensionality problems and avoid local optima (Kicinger et al., 2005; Rutten, 2010). Genetic Algorithm (GA) is natively implemented in Rutten's Galapagos in Grasshopper. Octopus is a plug-in for Grasshopper, which implements two multi-objective evolutionary algorithms: SPEA-2 in its original form and HypE from ETH Zürich (Vierlinger, 2013). In this research another type of biological system – a social system – is

Swarm Intelligence (SI), which is to be introduced. SI is based on the collective behaviours of individuals interacting with each other and their environment. Nevertheless, unlike GA, PSO has no evolution operators (e.g. crossover, mutation). Compared to other EC techniques the information sharing mechanism in PSO is simpler and particles tend to converge quicker in most cases(swarmintelligence.org). PSO is a very natural and visual technique as it relies on social interaction, which humans have experienced in their lifetime, as opposed to millennia of evolutionary change. Finally, PSO has a low number of parameters so it is easy to fit, tune and adapt to similar or new problem domains. Moreover, as the authors of PSO say "The method [PSO] is computationally inexpensive in terms of both memory requirements and speed" (Kennedy & Eberhart, 1995: 1942). PSO performance has been already shown in multi-objective daylight enhancement. As Aly and Nassar (2013: 3031) say: "The PSO algorithm was able to find a better solution faster than all other algorithms considered in GenOpt[1]". The second aim of research is application of a PSO approach in Grasshopper in order to enable designers to realize its revolutionary abilities as a design tool.

[1] GenOpt® (Generic Optimization Program) is developed by the Berkeley Lab, University of California with the following optimization algorithms implemented: Generalized Pattern Search algorithms, Particle Swarm Optimization algorithms, a hybrid global optimization algorithm (that uses PSO for the global optimization and Hooke-Jeeves for the local optimization), Discrete Armijo Gradient algorithm and Nelder and Mead's Simplex algorithm, Golden Section and Fibonacci algorithms(http://simulationresearch.lbl.gov/)

3. Evaluation method and PSO approach

3.1 Problem formulation

In generic problems there is commonly no knowledge on how to calculate an optimum answer. Furthermore, it is not clear "how to test a tentative answer for correctness", but "it is possible to compare two proposed solutions and select the more correct one" (Rutten, 2014), where:

- *Tensor* (the set of input parameters of the model) is defined by independent elements (e.g. size of windows, material properties)
- *Quality* is represented by a single numeric value (single–objective optimization)

Genetic solvers work with a fitness function, which is responsible for rating the quality of solutions (Rutten, 2014). To represent a generic design problem the following example of daylight availability is presented. A new office building is going to be constructed in a dense urban structure. Building codes specify the minimal number of hours of daylight availability and determine the Solution Space (black boundary in Figure 2). The building may be situated anywhere as long as its "centre" is within this boundary. The Quality of every possible Tensor (sets of x, y input parameters) is measured by the fitness function. The Quality is a single numeric value estimated as a total sum of the number of hours, when the sun rays catch the selected points in the facade (vertexes of mesh). The result for the random location of the building is shown in Figure 2 (left). The gradient on the facade exemplifies the measurement of daylight availability hours calculated for this position. For every Tensor the quality measure q is given and it is represented by vertical axis in Figure 2 (right).

All possible solutions create the points of the fitness landscape. Searching for the highest point of a Fitness Landscape is equivalent to searching for the maximum of the fitness function and finding the best position for the situation of the building in the plot.

3.2. Particle Swarm Optimization (PSO) - pseudo code and implementation

The benchmarking tests will be conducted on fitness landscapes to evaluate both the existing evolutionary solvers and a new proposed approach utilizing Particle Swarm Optimization algorithm. PSO is a population based stochastic optimization technique developed by Eberhart and Kennedy (1995). It was intended for simulating social behaviour (e.g. bird flocking or fish schooling). In a simplified version the algorithm is observed to perform optimization. In PSO the process is initialized with a group of random particles. All particles have assigned fitness values and velocities that direct them through the problem space by following the current optimum particles: *pbest* and *gbest*.

pbest is the best solution that has been achieved by the particle so far, while *gbest* is the best value obtained so far by any particle in the population. When those two values are known the position and velocity of each particle is updated with the following equations:

```
v[] = v[] + c1 * rand() * (pbest[] - present[]) + c2 * rand() * (gbest[] – present[])
```
(a)

```
present[] = persent[] + v[]
```
(b)

Shi and Eberhart (1998) introduced the internal weight factors to prevent velocities becoming too large. Numerous studies exist on adjusting internal tuning parameters. For the purposes of this research the following parameters that are commonly used were adopted: learning factors c1= 2 and c2 = 2 (Eberhart & Kennedy, 1995), internal weight factor w = 0.3. For the visualization purposes the maximum velocity (MaxVel*)* is limited to 0.15 and the number of particles is reduced to 30. Each seed is random with every optimization run.

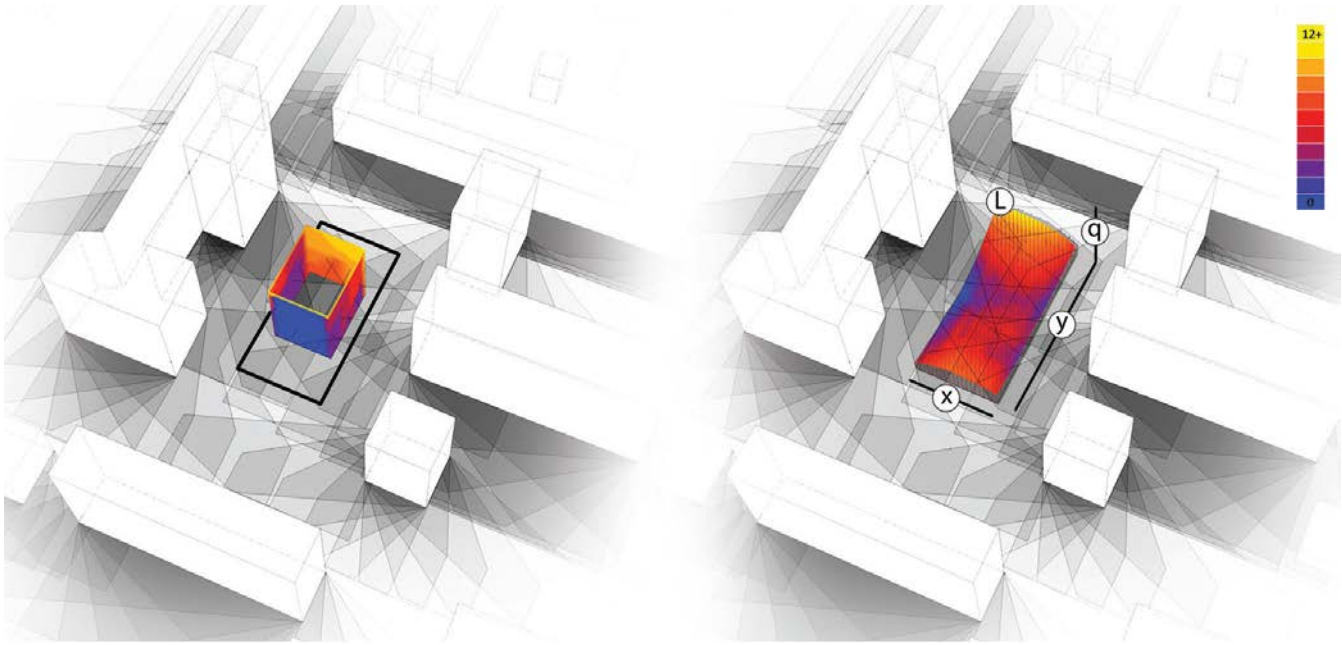

Fig 2: Optimization problem example, where: x, y –Elements of Tensor, q – Quality , L –Fitness Landscape of q; (yellow indicates a “high” and navy blue a “low” number of daylight availability hours at a point in facade).

4. Results

The presented tests are conducted on Fitness Landscapes as analogies to optimization problems. For visualization only two dimensional problems are considered, but PSO has been tested on problems with many features (Xue, Zhang, & Browne, 2013). For all tests the Runtime Limit is the termination criteria and it is set as 60 seconds per run. The solvers were employed with their default set of settings, while in the implemented PSO optimizer the standard settings were used. The results are presented both graphically in Figure 3 and numerically in Tables 1-4. The Q (Quality) is the quality value of the solution discovered by solver. T (Time) is the moment in a 60s run when the solver first discovers the best solution for the whole run and S (Sample) is the number of measurements, which indicates the extent of the search in 60s. Ten runs give statistical confidence in results as EC and SI can vary in the solution paths. From the first mapping results it is clear that Goat's mapping process strongly differs from the mapping of the other solvers. The orthogonal way of searching indicates that the implemented in Goat algorithm: *Global, evolutionary (CRS2)* is deterministic rather than stochastic. It is evident that Goat searches the smallest space, among all solvers (low density of the lines connecting sample points and low number of samples in Tables 1-4). Results also reveal the strong prevalence of Octopus and PSO Optimizer in all aspects of performance. Moreover, the PSO approach in all cases managed to discovered the best solutions in ten runs and proved to be faster than the alternatives.

4.1.1One solution

The results from the ten runs of evolutionary solvers for the problem with one solution are presented below (Table 1). Goat on average made only 294.5 samples while Galapagos executed 452.3 and Octopus 2516.2. The density of search has often a direct influence on the accuracy and quality of the discovered solutions. Goat did not discover the optima for the problem, when Octopus and PSO in every run yielded the optimal result and Galapagos was wrong only in the first run.

run	Galapagos			Goat			Octopus			PSO		
	Q	T	S	Q	T	S	Q	T	S	Q	T	S
1	2,09	53,68	361	2,19	32,23	283	2,25	20,48	2364	2,25	1,64	108930
2	2,25	43,57	431	2,24	24,08	319	2,25	13,70	2487	2,25	0,77	185940
3	2,25	35,41	471	2,15	35,06	255	2,25	15,10	2468	2,25	1,50	158310
4	2,25	41,29	465	2,10	58,95	229	2,25	19,12	2557	2,25	0,74	155130
5	2,25	41,29	438	2,24	45,24	435	2,25	17,97	2561	2,25	0,74	180030
6	2,25	34,51	452	2,19	59,37	284	2,25	11,91	2659	2,25	1,07	192600
7	2,25	43,61	432	2,21	3,09	194	2,25	18,07	2553	2,25	1,31	183690
8	2,25	32,24	508	2,19	48,93	401	2,25	19,63	2460	2,25	0,57	196770
9	2,25	29,55	469	2,23	58,95	287	2,25	12,30	2469	2,25	0,88	188610
10	2,25	21,53	496	2,10	30,00	258	2,25	7,31	2584	2,25	0,95	172350
Avr	**2,23**	**37,67**	**452**	**2,18**	**39,59**	**295**	**2,25**	**15,56**	**2516**	**2,25**	**1,02**	**172236**

Table 1: Results for optimization problem with one solution.

4.1.2 One high quality and 2 low quality solutions

run	Galapagos			Goat			Octopus			PSO		
	Q	T	S	Q	T	S	Q	T	S	Q	T	S
1	3,12	59,51	489	2,38	10,84	83	3,12	26,83	2344	3,12	1,13	215400
2	3,12	18,13	1135	1,89	48,46	104	3,12	34,64	2255	3,12	2,46	96900
3	3,12	16,45	1379	2,97	52,55	306	3,12	21,67	2273	3,12	1,15	158910
4	3,12	39,54	613	3,12	49,83	295	3,12	17,57	2285	3,12	1,34	165600
5	3,12	14,32	1349	2,43	11,28	117	3,12	21,73	2286	3,12	0,79	212640
6	3,12	19,25	1259	3,06	59,39	297	3,12	23,89	2288	3,12	2,65	95650
7	3,12	58,37	369	2,93	58,82	153	3,12	22,14	2379	3,12	1,08	205500
8	3,12	17,53	1222	2,82	10,80	100	3,12	16,05	2281	3,12	2,69	86580
9	3,12	16,42	1166	1,45	38,24	91	3,12	15,28	2286	3,12	2,15	178620
10	3,12	12,89	1196	2,52	23,24	111	3,12	28,41	2277	1,77	0,00	170970
Avr	**3,12**	**27,24**	**1018**	**2,56**	**36,35**	**166**	**3,12**	**22,82**	**2295**	**2,99**	**1,54**	**158677**

Table 2: Results for optimization problem with one high-quality and two low-quality solutions.

In the optimization problem with one high-quality solution and two low-quality solutions, Goat in the Run 2 and Run 9 became stuck in local optima. Galapagos in comparison to Octopus made less measurements, but still in every run managed to discover the high quality optima. Octopus had no errors, while PSO optimizer was only once stuck in low quality optima.

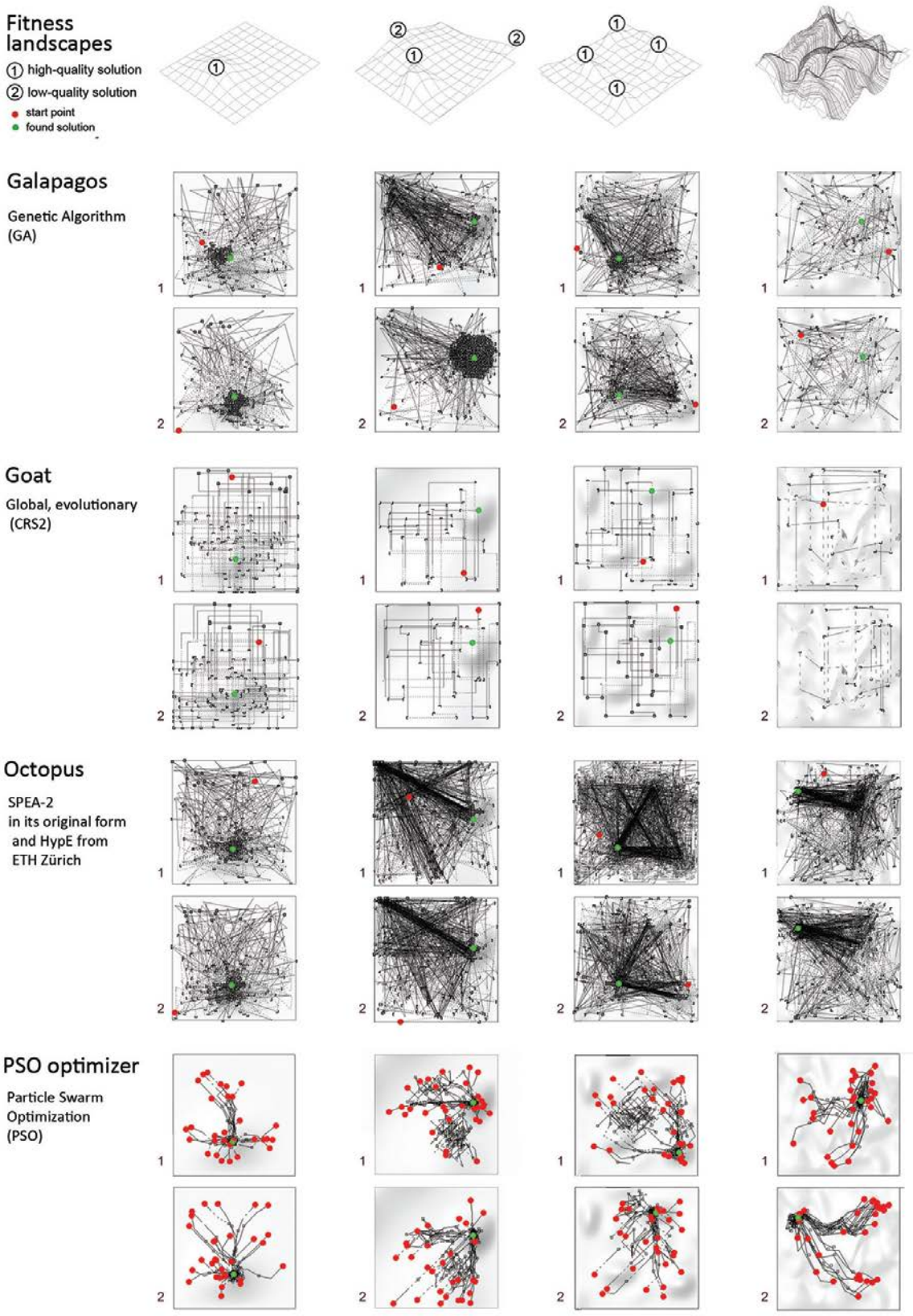

Fig 3: Visualization of mapping processes in single objective evolutionary optimization.

4.1.2 More than one optimal solution

The third optimization problem had four equal high-quality solutions to test the ability to discover equally good competing solutions. An interesting fact is that none of the solvers in all runs discovered the fourth peak, which is situated near the left border of the Fitness Landscape. This may suggest that for solvers it is easier to search in the middle of solution space. Another outcome from this series of runs is that neither Galapagos nor Octopus showed the tendency to find the closest to the start point peak with default settings, so that the initial input values of tensor elements do not influence the optimization outcomes. In multiple runs PSO managed to find three high peaks with average time below 1 s, however it did not always yield the best result in the domain.

run	Galapagos			Goat			Octopus			PSO		
	Q	T	S	Q	T	S	Q	T	S	Q	T	S
1	1,62	32,37	456	1,58	50,40	100	1,62	32,48	2088	1,60	0,57	192390
2	1,62	56,45	355	1,89	48,46	104	1,62	19,98	2078	1,62	1,38	196500
3	1,58	59,25	321	1,52	39,82	113	1,62	14,12	2078	1,62	0,45	230820
4	1,62	55,36	336	1,44	36,00	110	1,62	17,29	2191	1,62	2,15	120840
5	1,62	51,65	316	1,33	14,88	125	1,62	19,16	1893	1,60	1,07	239910
6	1,60	58,93	391	1,61	43,56	73	1,62	16,08	2111	1,62	1,69	109680
7	1,62	58,75	336	1,56	43,17	82	1,62	25,41	2080	1,62	0,46	233850
8	1,61	56,93	293	1,58	0,00	72	1,62	34,27	2083	1,60	0,88	217650
9	1,62	47,83	345	1,45	38,24	91	1,62	25,29	2144	1,62	0,54	206220
10	1,60	49,88	338	1,24	29,27	91	1,60	23,82	2102	1,62	0,39	230250
Avg.	**1,61**	**52,74**	**349**	**1,52**	**34,38**	**96**	**1,62**	**22,79**	**2085**	**1,61**	**0,96**	**197811**

Table 3: Results for optimization problem with more than one optima.

4.1.4 Bumpy Landscape

The results from the last optimization problem, with bumpy landscape, reveal the performance of solvers in a complex domain. During one minute Octopus went through about 20 generations, but Galapagos hardly managed to produce one generation. PSO made 6926 steps on average with high precision. Octopus yielded a high quality solution in most runs, but PSO showed the best performance in the 10-run series both in terms of accuracy and speed. It discovered on average better solution than Octopus.

run	Galapagos			Goat			Octopus			PSO		
	Q	T	S	Q	T	S	Q	T	S	Q	T	S
1	2,96	55,94	133	2,72	2,35	51	3,26	18,48	2088	3,02	0,75	239430
2	2,93	0,94	127	2,88	44,71	51	3,26	30,81	707	3,26	1,07	214560
3	3,15	25,85	123	2,76	34,47	47	3,26	17,79	2078	3,02	0,55	214410
4	3,14	51,56	64	2,68	37,30	37	3,26	14,57	2191	3,26	3,21	106470
5	2,86	15,79	57	2,67	12,00	40	3,26	32,20	1893	3,02	0,83	180210
6	2,85	47,65	68	2,50	0,00	40	3,26	17,28	2111	3,02	1,09	217560
7	2,85	0,00	1	2,31	27,00	40	2,89	14,02	2080	3,01	1,26	219810
8	2,36	22,11	57	2,04	45,88	34	2,89	26,62	2083	3,27	1,69	229002
9	2,52	30,00	64	2,13	51,63	43	2,89	21,35	2144	3,01	0,96	231510
10	2,46	4,29	70	2,21	18,57	42	2,89	25,72	2102	3,26	1,03	225030
Avg.	**2,81**	**25,41**	**76**	**2,49**	**27,39**	**43**	**3,11**	**21,88**	**1948**	**3,12**	**1,24**	**207799**

Table 4: Results for optimization problem with bumpy fitness landscape.

5. Discussion and guidelines

The presented tests show how the evolutionary solvers in Grasshopper perform in various optimization problems with the default set of settings. Better results might be achieved by adjusting internal tuning parameters to the specific optimization problems (Rutten, 2014; Shi & Eberhart, 1998). For EC in problems with numerous local optima, where the high quality solutions have small basins of attraction, the increase of population in settings is recommended. It may increase the time of computation, but dense sampling on the Fitness Landscape decrease the chances of missing the solutions with small basins of attraction (Rutten, 2014). Short distance sampling is predisposed to suffer from the tiny amount of numeric noise (Rutten, 2014), but in multiply runs, there is greater chance not to miss similar high-quality solutions. For the problems with multiple equal solutions it is recommended to run the optimization process several times with random start points in order to discover more than one optimal solution. As results show the speed efficiency of PSO let the users run the

optimization process several times in the same time as one run with EC-based solvers. Consequently, in problems with more than one optimal solutions, users have greater chance to discover all of them. For visualization purposes the maximum velocity of particles in PSO optimizer was limited. Thus PSO may be able to achieve similar results in shorter runs. PSO, like EC techniques, is a gradient-free method and does not require an optimization problem to be differentiable. Employing PSO, architects and planners do not have to be aware of technical details while constructing the optimization problem. Therefore it can be successfully used on the complex optimization problems, which are increasingly common in architectural practice.

5. Conclusions

Visualizations reveal that differences in the EC and SI mapping processes affect the quality of discovered solutions and the time taken. Substantial difference in the performance of the techniques was only observed in the complex problem. In the presented tests, the introduced PSO technique that relies on SI rather than EC confirmed its computational speed, intuitiveness and robustness in complex optimisation problems. Such problems are increasingly common in modern architectural practice, so PSO has the potential to become a revolutionary design tool.

6. Acknowledgments

The pseudo-code was adopted from http://www.swarmintelligence.org/tutorials.php. The input of Agata Migalska and Thomas Lorns in implementation of PSO procedure is appreciated.

7. References

Aly M., Nassar, K., 2013. Integrating performance and parametric design tools for urban daylight enhancement. Proceedings of BS2013: 13th Conference of International Building Performance Simulation Association, Chambéry, France, August 26-28

Cichocka, J.M., Musikhina, E.A., 2014. Methods of Optimization in Architecture. Research Journal of International Studies, №1 (20) 2014 Part 3, pp. 109-111

Kennedy, J., Eberhart, R., 1995. Particle Swarm Optimization. Proceedings of IEEE International Conference on Neutral Networks IV, pp. 1942-1948

Kicinger, R., Arciszewski, T. and De Jong K. 2005. Evolutionary Computation and Structural Design. Survey of the State of the Art. Journal of Computers and Structures. Volume 83, Issue 23-24, pp. 1943-1978.

Martyn, D. 2009. Rhino Grasshopper.AEC magazine. vol. 42.

Rutten, D. 2010. Grasshopper - Computing Architectural Concepts. Lecture on 21.09.2010 at the conference Advances in Architectural Geometry, Vienna, Austria, Sept. 18-21, 2010.

Rutten, D. 2014. Navigating multi-dimensional landscapes in foggy weather as an analogy for generic problem solving. Proceedings of the 16th International Conference On Geometry And Graphics 2014. Innsbruck, Austria, August 04 - 08, 2014.

Shi, Y., Eberhart, R.C., 1998. A modified particle swarm optimizer. Proceedings of IEEE International Conference on Evolutionary Computation. pp. 69–73.

Vierlinger, R. 2013. Multi Objective Design Interface. Master Thesis. University of Applied Arts Vienna

Xue, B., Zhang, M., Browne, W. N. L., Particle Swarm Optimisation for Feature Selection in Classification: A Multi-Objective Approach, IEEE Transactions on Cybernetics, 43, 6 (2013), pp. 1656-1671