HYDRA

Distributed Multi-Objective Optimization for Designers

Marcin Kosicki, Marios Tsiliakos, and Martha Tsigkari
Foster + Partners, 22 Hester Road, SW11 4AN, London, UK

Abstract. Architectural design problems can be quite involved, as there is a plethora of – usually conflicting – criteria that one has to address in order to find an optimal, performative solution. Multi-Objective Optimization (MOO) techniques can thus prove very useful, as they provide solution spaces which can traverse the different trade-offs of convoluted design options. Nevertheless, they are not widely used as a) they are computationally expensive and b) the resulting solution space can be proven difficult to visualize and navigate, particularly when dealing with higher dimensional spaces. This paper will present a system, which merges bespoke multi-objective optimization with a parametric CAD system, enhanced by supercomputing, into a single, coherent workflow, in order to address the above issues. The system architecture ensures optimal use of existing compute resources and enables massive performance speed-up, allowing for fast review and delivery cycles. The application aims to provide architects, designers and engineers with a better understanding of the design space, aiding the decision-making process by procuring tangible data from different objectives and finally providing fit (and sometimes unforeseen) solutions to a design problem. This is primarily achieved by a graphical interface of easy to navigate solution spaces of design options, derived from their respective Pareto fronts, in the form of a web-based interactive dashboard. Since understanding high-dimensionality data is a difficult task, multivariate analysis techniques were implemented to post-process the data before displaying it to end users. Visual Data Mining (VDM) and Machine Learning (ML) techniques were incorporated to facilitate knowledge discovery and exploration of large sets of design options at an early design stage. The system is demonstrated and assessed on an applied design case study of a master-planning project, where the benefits of the process are more evident, especially due to its complexity and size.

Keywords: Distributed Computing, Parallel Computing, Performance Design, Optimization; Evolutionary Computing, High Performance Computing

1 Introduction

According to Keough and Benjamin (2010) the challenge of the architect is to create a high performing building design that is the result of often competing objectives. This desire for performance driven design based on variable criteria has led designers to introduce a variety of computational techniques (used extensively and for many years in other industries) in the design process, such as multi-objective optimization (MOO), distributed computing and data visualization. In engineering, architecture and product

design, optimization is often tied to simulation software such as Finite Element Analysis (FEA) (Kicinger, Arciszewski and DeJong, 2005) and there has been a lot of bespoke efforts to develop systems that allow for incorporation of simulation engines in the optimization process. The problem that is then posed is that of speed: simulations such as FEA, Computational Fluid Dynamics (CFD), Daylight etc. require long run-times, even for small models. When that is scaled up for every analysis and for thousands of populations of a few hundred individuals each (a typical setup for an optimization algorithm), the computation time can be significant and even prohibited, particularly when dealing with fast passing projects and urban scale models. Roudsari, Yi and Drew (2012) tried to tackle this problem by utilizing shared network system to speed up daylight autonomy simulation using Radiance. Kyropoulou, Ferrer and Subramaniam (2018) distributed annual daylight simulation over Microsoft Azure cloud computing system. Both applications were oriented towards a single domain of daylight simulation using Daysim and Radiance where only the raytracing part of the simulation was distributed. The systems used Rhino and Grasshopper as an interface for generating simulation data and visualization of the result. In parallel, CAD software companies, such as Autodesk (Project Refinery) and McNeel (RhinoCompute) have not only been developing optimization tools but are also moving towards cloud-based solutions with custom APIs, enabling the execution of computationally intensive tasks.

The authors of this paper, who are members of the Applied Research + Development group at Foster + Partners, will present a process developed to address the above issues: a system that runs a bespoke MOO analysis within a CAD system, using custom simulation engines and enhanced by supercomputing, capable of converging orders of magnitude faster than off-the-shelf software. The paper will initially focus on the distributed MOO process developed, presenting its system's architecture and the way parallelization is achieved. Consecutively, the authors will showcase how the results of the MOO are presented through a custom design space exploration interface. Finally, the process is demonstrated via a case study. The presented approach is assessed in comparison to previous multi-optimization techniques for the design industry, in terms of the range of the design space and convergence rates. Multi-dimensional Data Visualization techniques such as Self Organized Maps (SOMs) and hierarchical clustering are also compared to existing studies. Finally, similar approaches in generative model development using discrete tiling, L-systems or brute force techniques are examined in parallel to the documented case study.

2 Distributed Multi-Objective Optimization

2.1 Optimization as Design Space Exploration Tool

The process of architectural design often involves challenging optimization problems in which there are multiple and often conflicting objectives that must be simultaneously satisfied (Newton, 2018). For a conceptual Multi-Objective Optimization (MOO) process to gain traction with designers looking for creative, expressive forms, it must yield a diverse range of high-performing results that meet a variety of aesthetic preferences

(Brown, Tseranidis, & Mueller, 2015). According to Mueller and Ochsendorf (2013), these results must be generated and evaluated rapidly, while balancing allowances for designer preference with clear guidance towards the best solutions. Although multi-objective optimization design problems are computationally demanding and have slow convergence times, supercomputing has not been widely used in the context of architectural optimization. This is mostly because contemporary parametric CAD software such as Grasshopper for Rhino or Dynamo for REVIT have been mainly built and optimized to run efficiently on a single workstation rather than in a cloud.

2.2 Precedent Work

Currently the most popular evolutionary optimization frameworks in architectural industry are based on plugins for Grasshopper, namely Galapagos (Rutten, 2013) and Octopus (Vierlinger, 2013). Both plugins were designed to work with a single instance of Rhinoceros, which is a significant bottleneck, as it limits their applicability to simple optimization studies excluding most of the real-world problems. Mueller (2015) built a prototype of an evolutionary optimization framework using Generative Components from Bentley. The system was capable of distributing phenotype and fitness calculations using Microsoft Azure. Chaszar, Buelow and Turrin (2016) built a design space exploration system called ParaGen based on a Non-Destructive Dynamic Population Genetic Algorithm. In this system all solutions were maintained in a database and could be recalled or searched at any time by a designer. The system used an SQL database and a web interface to visualize design space. Hydra conceptually utilized the ideas from Muller, choosing however Grasshopper as the main geometry generation tool and an interface to performance simulation engines. This significantly improved the overall robustness of the system and allowed to tackle much more elaborate design problems than the ones analyzed using ParaGen.

2.3 Methodology

Parallelization. The basic optimization requires evaluation of fitness values for candidate solutions (individuals). In a real-world scenario huge computational time is usually required to evaluate each individual. In this situation, it is typically impossible to obtain a certain result in a reasonable calculation time. To solve this issue, a parallel calculation is often adopted. The most robust implementation of parallelization is known as a controller-worker model or global parallelization (Branke, Schmeck, Deb, & Reddy S, 2005) (Van Veldhuizen, Zydallis, & Lamont, 2003). In this scenario, a controller application running on a master node is responsible for initialization, crossover, mutation and selection except for evaluation of individuals. In evolutionary computation, multiple individuals exist in a population. The evaluations are completely independent from each other and could be carried out on different worker nodes. The controller node generates a population of initial solutions and distributes individuals to independent worker nodes. Then, the workers simultaneously evaluate all individuals and the fitness values are collected by the controller node. Based on the fitness values, the controller

identifies promising individuals and generates new individuals by applying genetic operators. This is repeated until a given termination condition is met (Talbi et al., 2008).

Inter-process communication. Recent developments in cloud computing and software interoperability technologies greatly simplify the task of building and managing massively parallel computing infrastructure. The widespread adoption of Representation State Transfer (REST) software architecture pattern (Fielding, 2000) for Web services proved to be especially successful. REST is based on stateless transactions. Software components which implement such transactions can be freely redeployed if something fails, and they can scale to accommodate load changes. This is because any request can be directed to any instance of a component; there can be nothing saved that has to be remembered by the next transaction. This makes its especially useful in cloud computing scenarios.

2.4 System Architecture

By implementing the REST pattern, the authors were able to develop a bespoke system named Hydra which is scalable, can be adapted to any programmable CAD software and can leverage both on-premises HPC (High Performance Computing) and cloud solutions. The system uses a Microservice software pattern (Newman, 2015) and a REST API for inter-process communication. The system's back-end, responsible for storing and querying metadata, was built on top of an SQL database. The front-end for data visualization and user interaction was implemented as a web application. To leverage the power of distributed computing, a multi-objective GA was decoupled from a CAD system and developed as a separate standalone application (controller). The algorithm was based on Strong Pareto Evolutionary Algorithm 2 (SPEA2) adopted from the JMetal framework (Durillo & Nebro, 2011). The controller application only operates on the metadata level using genome-score vector pairs. It uploads and retrieves each individual's metadata from the SQL database and sends job request to an on-premises computer cluster. The cluster is controlled by a commercially available compute management system called Deadline from Thinkbox. A job in this context is understood as a set of customized definitions with bespoke components processed in parallel using worker nodes. Each definition pulls a respective genome from the database and creates a phenotype based on a predefined parametric model. Then the phenotype is evaluated using simulation engines and the aggregated scores are reported back to the database via a bespoke REST client. Additionally, both the phenotype mesh and partial simulation results are saved to the hard drive for future visualization. Grasshopper for Rhino was the environment chosen to take advantage of custom simulation engines previously developed in-house as well as community supported plugins such as Ladybug (Roudsari, Pak, & Smith, 2013). This approach limited data transfer between different simulation engines to a minimum and simplified interoperability issues. After all evaluations have finished, the compute management system notifies the controller application that an entire population has been processed and the next could be generated

(**Error! Reference source not found.**). This solution proved to be efficient and reliable.

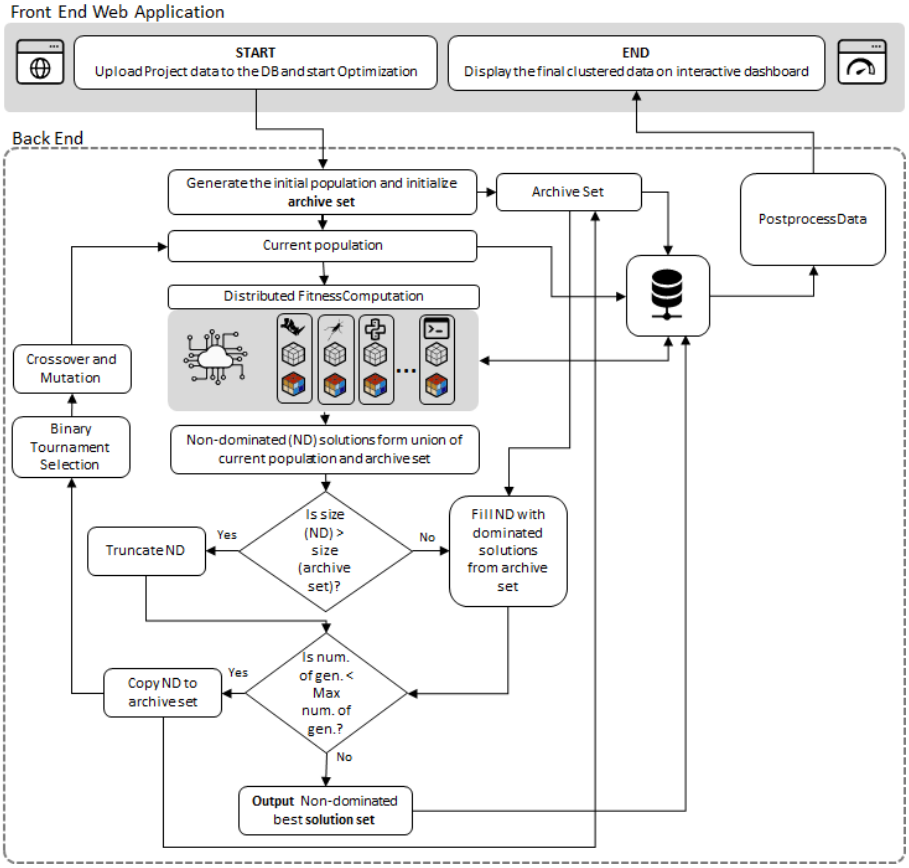

Fig. 1. High level diagram of the system

3 Case Study

3.1 Design Problem

The authors examined the robustness of the system on a highly complex real-life design case study. It involved a large master-plan proposal, with set planning regulation constraints and an open building brief. Hydra was introduced at the concept design stage, to help the team explore a vast number of generative massing possibilities based on their performance on set objectives. The entire process was broken down into 3 distinct parts: a) the urban model generation, both genotype and phenotype, which plugged into

our GA system, b) the custom simulation set-up based on the design team' s performance objectives and c) the parallelization of the entire process through Hydra.

3.2 Procedural City Generation

On the geometry generation side, an interpretation of a Procedural City Generation (PCG) model was introduced. The rapid pace of the project and the specifics of the constraints and program rendered the creation of a fast, flexible but at the same time robust system imperative for the conveyance of a beneficial exploration and evaluation of the design space.

Road network and land parcels. The PCG model employed a hierarchical sequence of actions for the generation of the urban massing, similar to the system architecture introduced by Parish and Müller (2001) for CityEngine. These steps start from the roadmap graph initially describing the allotments of available area, further subdivision of which, is achieved by introducing the anticipated common areas and pedestrian routes. This network consequently specified the available parcels for the building geometry distribution, which are then populated with potential building volume insertion locations.

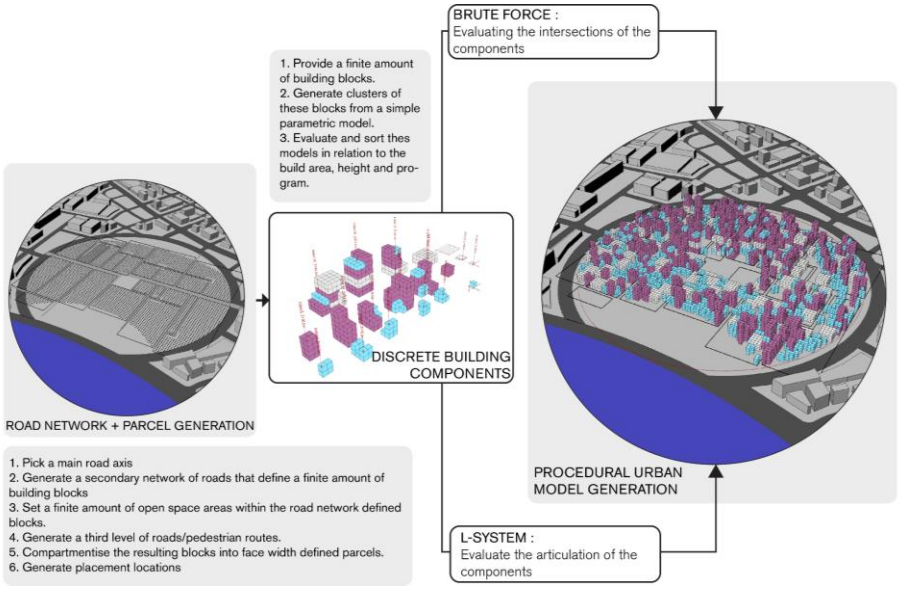

Fig. 2. Procedural City Model

In order to occupy these insertion coordinates with geometry, a series of discrete building components and their relevant connectivity in terms of program have been provided as modular blocks for the design. Our approach differs from the previously mentioned CityEngine one, due to the fact that the Level of Detail (LOD) is inherent to the geometry of the components, rather than being created in a sequence of operations eroding

the bounding box of a building volume. Moreover, this methodology does not employ a stochastic grammar to create mass models that fill out a specific parcel with randomized ranges of dimension (Muller et al., 2006) but enables a richer language of building volumes where components can interlock, avoid each other and cohere to density, height and boundary as numerical parameters depended on a primitive building voxelized module of certain dimensions. The purpose of this approach is not to generate accurate building volumes, nor to define tectonic characteristics. The resulting building mass is a flexible, rich in information model which will provide the input for the simulations set-up in order to evaluate it.

Building blocks population. Two different directions were explored for the population of the parcels. One utilizing an L-system (Prusinkiewicz & Lindenmayer, 1991) configuration and a second one using a brute-force placement of the blocks. The L-System is iteratively evaluating the neighboring or linked building blocks regarding their respective program connectivity starting from a randomly picked insertion point location from the aforementioned in the previous step. Whilst this methodology might be more consistent with a realistic space planning approach, it was by far more computationally intensive, as every new block would have to check at least 8 potential neighbors. In addition, due to the early stage of the project and the lack of an amalgamated program definition, it was challenging to define concise rules for the L-System parameters. The brute force placement method was preferred in the final simulation being significantly faster and at the same time providing a wider variety of massing assemblies hence a richer exploration of the design space. This exhaustive search of available placement locations and conglomerations of the voxelized components would run continuously until certain criteria, such as density, total height or availability of open spaces, were met. The building volume aesthetics are depended on the basic module dimensions and the combinations of this to form program as defined by the design team, thus a different module would lead to potentially less segregated mass. However, the emanated geometry is to be used solely to indicate which combination of building volume and urban space performs better, and not to suggest architectural solutions.

Gene Conversion. This modular and flexible volumetric system further facilitated a versatile parametrization of the three-dimensional metrics of space to a gene conversion. Rather than explicitly encoding the parameters of the buildings generated as individual genes, genes that control the generative process itself were instigated. The main reason behind this is that of parametrizing the process using a smaller number of genes, hence a smaller quantity of dimensions to explore. However, the hierarchical design of the PCG allowed more control over the final design, even with a just a few initial genes, resulting in greater complexity, due to the iterative population of the building components (**Error! Reference source not found.**). As such, the initial road network graph was characterized by a series of genes, the open urban spaces by others and number of available parcels by another gene. As a nested level of hierarchy each parcel had its own internal encoding, the dictated density, height, initial point for the volume population and available program.

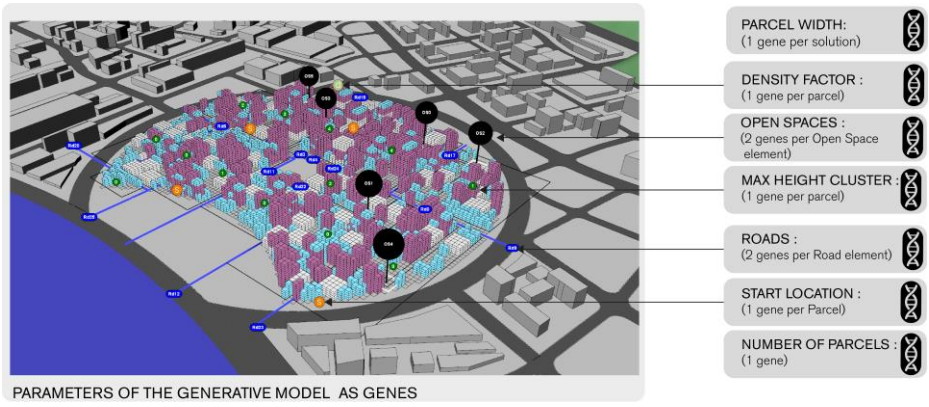

Fig. 3. Genotype Breakdown

3.3 Performance Evaluation

After the iterative population of elements has finished, the resulting elements are further voxelized in correspondence to the preferred grid module, which time is a finer definition of the initial building volume. The whole model can then be fed to the relevant simulation engines. A range of objectives were set for this project, including area, quality of view, environmental and spatial criteria. Floor Area Ratio (FAR)/Gross Floor Area (GFA) targets were derived directly from the voxels. Views of the waterfront and hills neighboring the site were analyzed, as well as cumulative solar radiation at street level and daylight potential, visual connectivity from the open spaces and walkability of the scheme. (see **Error! Reference source not found.**).

The FAR/GFA target is a simple addition of the area data associated with each voxel. The view analysis, on both targets, was conducted via a custom written analysis engine. Faces with no views are marked with black color whereas the viewing score is also displayed by the intensity of the color on the respective face. For instance, dark blue indicates greater view potential towards the sea, when lighter tones specify smaller values. Similarly, a graph-based system was utilized for the visual connectivity and walkability performance criteria, where the color gradient represents distance and time metrics. Finally, the Ladybug add-on was used for the Solar radiation and Skylight potential. All the routines were executed through the Grasshopper3d UI, limiting all the computation within one CAD framework.

After all the simulations are completed, individual performance scores and overall fitness values are fed back to the user and can be executed locally on a single node. Hydra, facilitates the parallelization of both the PCG process and the simulation tasks, sending back to the SQL database all the relevant metrics and scores whilst visually documenting each solution.

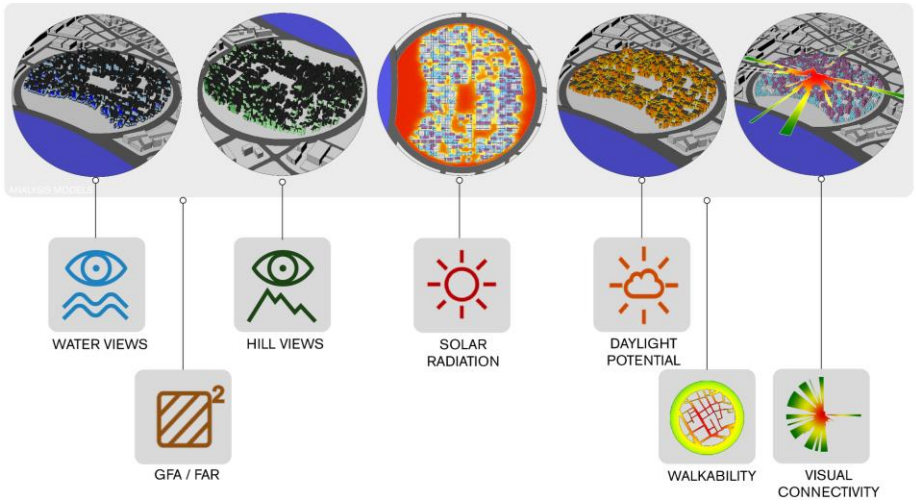

Fig. 4. Performance Driven Objectives

4 Results

4.1 Summary

Once the system was set-up the tasks were distributed to eight computing nodes, running five instances of Rhino in parallel, each containing in total 320 CPUs. In this case study over 6000 masterplan solutions were generated and evaluated over 54 hours. Figure 5 shows the comparison of the processing times per generation. The average processing time per individual (including phenotype generation and performance simulations) was 15.03 minutes. Using Hydra, the actual processing time per generation containing 60 individuals was on average 32.8 minutes. The total simulation time for 100 generations was 55 hours. If Hydra had not been used and the individuals had been processed sequentially, the same study would have taken 1504 hours (almost 63 days). This is illustrated by the Total Combined Processing time which shows linearly aggregated processing times for all individuals in given generation. Therefore, including wasted time due to the overhead of switching between individual jobs which are expensive to start, Hydra provided a 27x times speedup. The sudden spike in actual processing times around generation 33 was caused by a sudden influx of different jobs to the render farm, which temporarily reduced computational resources available for this study.

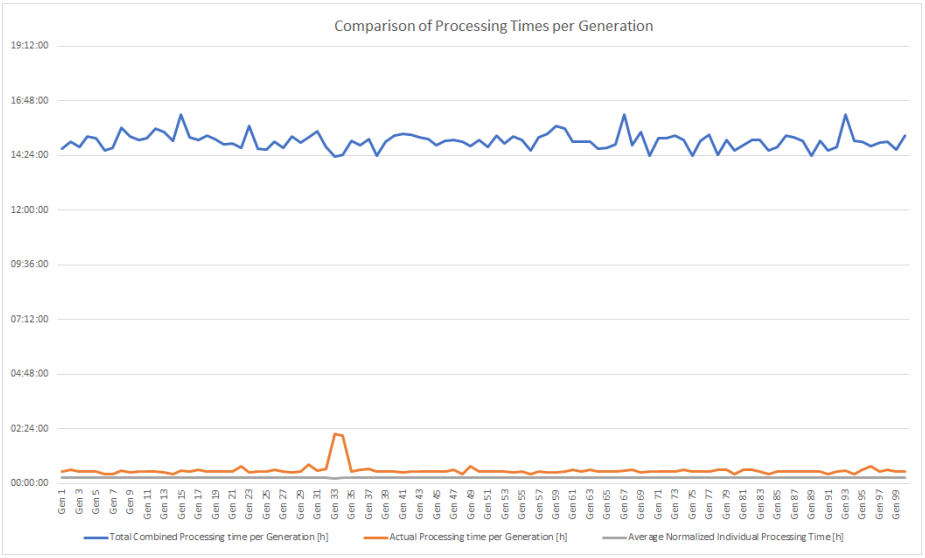

Fig. 5. Comparison of Processing Times per Generation

4.2 Interactive Dashboard

The results were then post-processed and visualized using an interactive web-based dashboard (see **Error! Reference source not found.**). The dashboard used multivariate analysis algorithms for exploring and understanding the relations between various design parameters, as self-organizing maps (Harding, 2016), hierarchical clustering and dendrograms (Vesanto & Alhoniemi, 2000) and parallel-charts (Sileryte et al. 2016), (Chaszar, Buelow and Turrin (2016). The detailed description of the dashboard is beyond the scope of this paper. These results were well received by the design team, primarily because of the fast delivery pace, but more importantly because they provided feasible design options, a coherent evaluation of the local, site-specific, design space and enabled assessing tradeoffs between conflicting objectives.

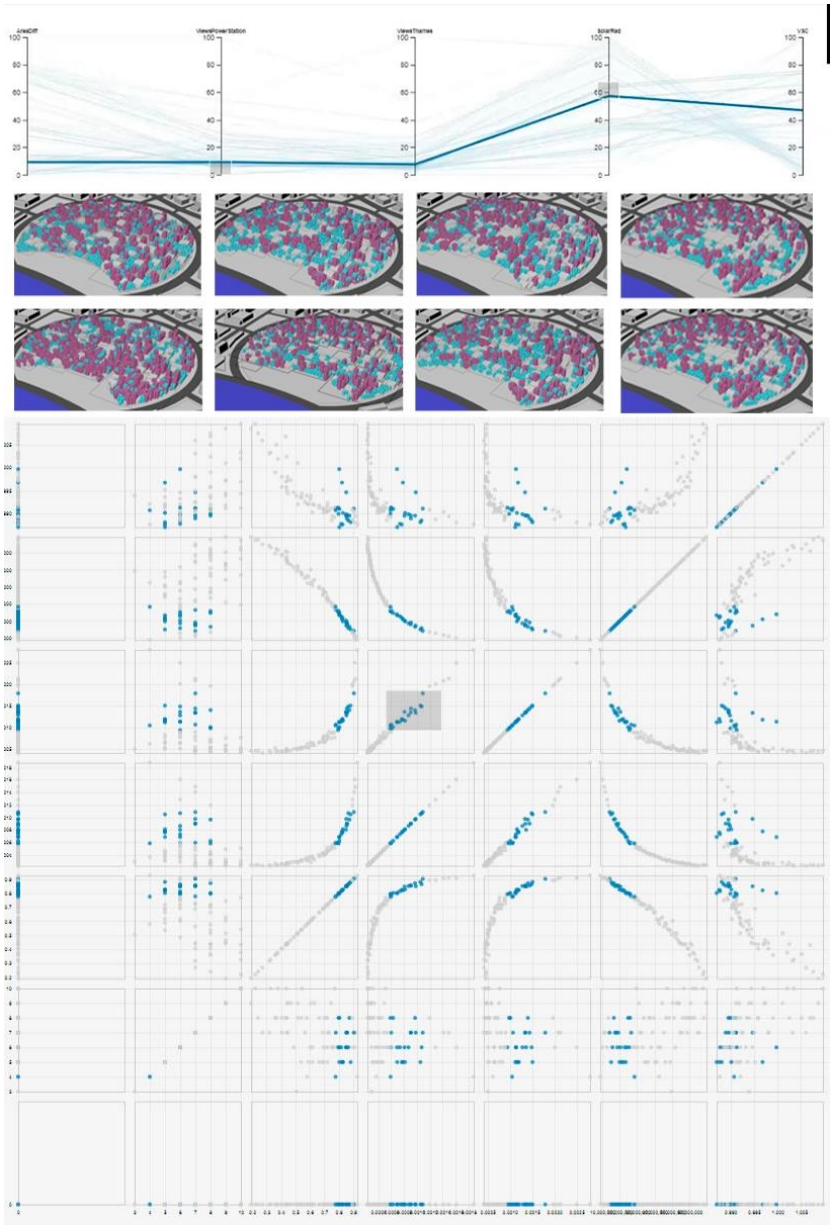

Fig. 6. Interactive Dashboard

5 Conclusions

This paper presented Hydra: a distributed multi-objective optimization system which was evaluated on a complex urban case study developed by members of the Applied Research + development group at Foster + Partners. The principles of Microservice software architecture applied to parametric CAD software and various performance analysis engines proved to be highly efficient and resilient. The system scaled up well, utilizing hundreds of CPUs. Hydra was capable of generating not only thousands of masterplans but also of running complex analysis on them (e.g. daylight potential, quality of view, solar radiation) considerably faster than currently widespread methods. It is a significant improvement over the previous case studies, which were limited either simplistic massing models or simulations only form a single performance domain. Moreover, it also proved that such design space exploration system, while combined with distributed computing platforms, could be effectively used at the early design stage even for complex models. Additionally, faster processing times and database storage capacity, both addressed by Hydra, allow for generation of massive data sets which are essential for building more sophisticated design systems based on predictive models. Current advancements in Machine Learning, especially in Convolutional Neural Network (CNNs) have been explicitly possible due to both access to large data sets and an increase in parallel compute power. This suggests that Hydra-like systems have a great potential to help advancing the state of the art in AEC computing

References

Branke, J., Schmeck, H., Deb, K., & Reddy S, M. (2005). Parallelizing multi-objective evolutionary algorithms: cone separation. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)* (pp. 1952–1957). IEEE. https://doi.org/10.1109/CEC.2004.1331135

Brown, N., Tseranidis, S., & Mueller, C. (2015). Multi-objective optimization for diversity and performance in conceptual structural design. In *Proceedings of the International Association for Shell and Spatial Structures (IASS), Future Visions, 17 - 20 August 2015, Amsterdam, The Netherlands*. Retrieved from http://digitalstructures.mit.edu/files/2015-09/ncb-iass-paper-final.pdf

Chaszar, A., Buelow, P. Von, & Turrin, M. (2016). Multivariate Interactive Visualization of Data in Generative Design Multivariate Interactive Visualization of Data in Generative Design. In A. Ramtin, A. Chronis, S. Hanna, & M. Turrin (Eds.), *SimAUD*. London.

Durillo, J. J., & Nebro, A. J. (2011). jMetal: A Java framework for multi-objective optimization. *Advances in Engineering Software*, *42*(10), 760–771. https://doi.org/10.1016/j.advengsoft.2011.05.014

Fielding, R. T. (2000). *Architectural Styles and the Design of Network-based Software Architecturese*. University of California, Irvine. Retrieved from http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

Harding, J. (2016). Dimensionality Reduction for Parametric Design Exploration. In S. Adriaenssens, F. Gramazio, M. Kohler, A. Menges, & M. Pauly (Eds.), *Advances in*

Architectural Geometry 2016 (pp. 204–221). Zurich, Switzerland: vdf Hochschulverlag AG. https://doi.org/10.3218/3778-4_19

Mueller, C., & Ochsendorf, J. (2013). An Integrated Computational Approach for Creative Conceptual Structural Design. *Proceedings of the International Association for Shell and Spatial Structures (IASS) Symposium 2013*, 1–6.

Mueller, V. (2015). Second Generation Prototype of a Design Performance Optimization Framework, (April).

Newman, S. (2015). *Building Microservices* (1 Edition). O'Reilly Media.

Newton, D. (2018). Multi-Objective Qualitative Optimization (MOQO) in Architectural Design. In A. Kepczynska-Walczak & S. Bialkowski (Eds.), *Computing for a better tomorrow - Proceedings of the 36th eCAADe Conference* (Vol. 1, pp. 187–196). Lodz, Poland.

Parish, Y. I. H., & Müller, P. (2001). Procedural Modeling of Cities. *28th Annual Conference on Computer Graphics and Interactive Techniques*, (August), 301–308. https://doi.org/10.1145/383259.383292

Prusinkiewicz, P., & Lindenmayer, A. (1991). *The Algorithmic Beauty of Plants*. Springer-Verlag.

Roudsari, M. S., Pak, M., & Smith, A. (2013). Ladybug: a Parametric Environmental Plugin for Grasshopper To Help Designers Create an Environmentally-Conscious Design. *13th Conference of International Building Performance Simulation Association*, 3129–3135. Retrieved from http://www.ibpsa.org/proceedings/bs2013/p_2499.pdf

Rutten, D. (2013). Galapagos: on the logic and limitations of generic solvers. *Architectural Design 83(2)*, 132–135.

Sileryte, R., D'Aquilio, A., Di Stefano, D., Yang, D., & Turrin, M. (2016). Supporting Exploration of Design Alternatives using Multivariate Analysis Algorithms. In A. Ramtin, A. Chronis, S. Hanna, & M. Turrin (Eds.), *Proceedings of the Symposium on Simulation for Architecture and Urban Design* (pp. 215–222). London, UK.

Talbi, E. G., Mostaghim, S., Okabe, T., Ishibuchi, H., Rudolph, G., & Coello Coello, C. A. (2008). Parallel approaches for multiobjective optimization. In *Multiobjective Optimization* (Vol. 5252 LNCS, pp. 349–372). https://doi.org/10.1007/978-3-540-88908-3-13

Thinkbox. Deadline. Retrieved April 14, 2019, from https://deadline.thinkboxsoftware.com

Van Veldhuizen, D. A., Zydallis, J. B., & Lamont, G. B. (2003). Considerations in engineering parallel multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, *7*(2), 144–173. https://doi.org/10.1109/TEVC.2003.810751

Vesanto, J., & Alhoniemi, E. (2000). Clustering of the self-organizing map. *IEEE Transactions on Neural Networks*, *11*(3), 586–600. https://doi.org/10.1109/72.846731

Vierlinger, R. (2013). A Framework for flexible search and optimization in parametric design. *Rethinking Prototyping - Proceedings of the Design Modelling Symposium*, (October 2013). https://doi.org/10.13140/RG.2.1.1516.8727