

PARENG 2017: The 5th International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering, Pécs, Hungary, 30-31/5/2017.

Single-branch Truss-Z Optimization Based on Image Processing and Evolution Strategy

M. Zawidzki, J. Szklarski
Institute of Fundamental Technological Research of the
Polish Academy of Sciences, Warsaw, Poland

Abstract

Truss-Z (TZ) is a skeletal system for creating free-form pedestrian ramps and ramp networks among any number of terminals in space. TZ structures are composed of four variations of a single basic unit subjected to affine transformations (mirror reflection, rotation and combination of both).

This paper presents a new approach to the optimization of the layout of a single-branch Truss-Z (STZ) in constrained environment (E). The problem is formulated as follows: create an STZ from a start (sP) to end point (eP) without self-intersections and collisions with two obstacles. This is a multi-criterial optimization problem where three independent objectives are subjected to minimization: the total number of modules (n), the “reaching error” to eP (ϵ_r) and the “overlapping error” (ϵ_o). All three criteria are **weighted** and aggregated to a single cost function (CF).

The calculation of CF is based on image processing of rendered geometry of individual STZs in E . The optimization is performed by population-based classic heuristic method - Evolution Strategy (ES). The computation of CF is the most time-consuming, however, its parallelization is rather straightforward.

Two parallelization methods are presented: distribution over Wolfram Lightweight Grid™ and application of general purpose graphical processing units (GPGPUs) with the use of CUDA platform.

Keywords: Extremely Modular System, Truss-Z, discrete optimization, image processing, rasterization, GPGPU, CUDA, *Mathematica*™, Wolfram Lightweight Grid™.

1 Introduction

The idea of Extremely Modular System (EMS) has been introduced in 2016 [1] by the author. The purpose of EMS is to create free-form objects in a given environment (E)

without obstacle-, and self-collisions. The concepts of two EMSs: Truss-Z (TZ) and Pipe-Z (PZ) have been presented by the author in References [3] and [2], respectively.

EMS jointly meets the following three criteria:

1. EMS allows for creation of structurally sound free-form structure.
2. **The number of module types in EMS is minimal.**
3. EMS is not constrained by a regular tessellation of space.

PZ is more “fundamental” and forms spatial single-branch *knot-like structures* by assembly of one type of module, called PZM. TZ is a skeletal system for creating free-form *pedestrian ramps* and *ramp networks* among any number of terminals in space.

2 Truss-Z

TZ structures are composed of four variations of a single basic module (R) subjected to affine transformations (mirror reflection, rotation and combination of both). The naming convention follows the *right-hand-rule*, and R stands for a unit which “turns left and ascends”. Unit L (left) is a mirror reflection of unit R . By rotation, they can be assembled in two additional ways (R_2 rotated R and L_2 rotated L), effectively giving four types of units. Some examples are shown in Fig.1.

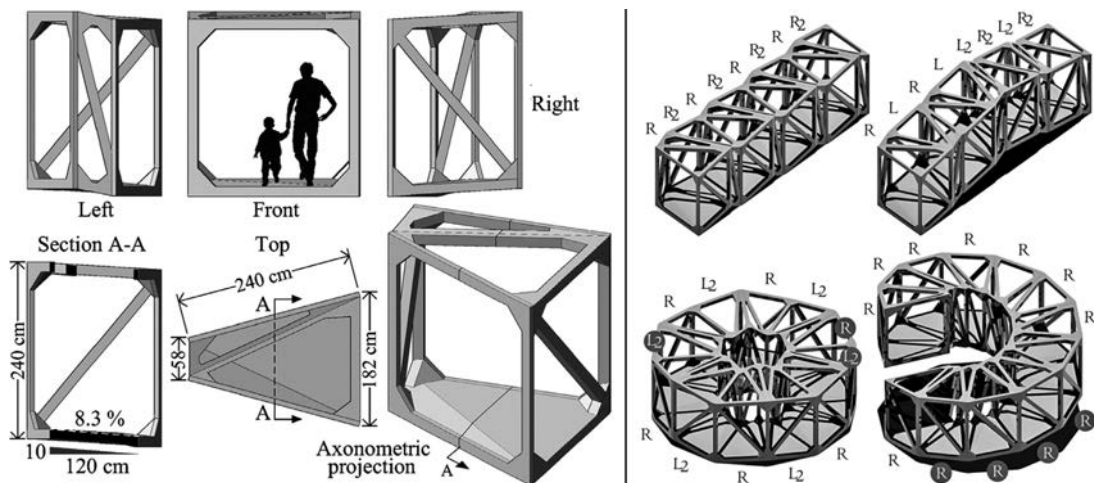


Figure 1: On the left: TZ basic unit (R). Upper row: orthographic views. Lower row: section A-A showing the slope, top view and axonometric view. On the right: some basic examples of single-branch TZ structures. Upper row: “straight and flat” with 8 units, “straight up & down” (8). Lower row: a flat ring (12), and a helix (12).

Previous research on TZ system focused on discrete topological and geometric optimization of Truss-Z module (TZM) [5] and a number of methods (including graph-theoretic algorithms [6], meta-heuristic [7], etc.), for global optimization of sample TZ structures, have been presented. Moreover, primary analysis of deployable TZM has been presented in [8].

As mentioned in the Introduction, all EMS structures, including TZ must be installed in an environment E without violating given obstacles (O) and without self-intersections. However, in the previous studies, the environment setups made possible to ignore the self-intersection prohibition. In all those cases it was intuitive to assume that the best TZs, that is comprised of the smallest number of units would not self-intersect. It was natural, as intersection detection is a well known and very difficult problem in surface modeling [9], especially that each TZM is considered as an individual graphical element to be checked against each other. The computations are performed in *Mathematica*TM environment.

3 The problem

This paper introduces the problem of self-intersection to the optimization of a two-dimensional single-branch TZ path (TZP). A specifically arranged environment E^* contains two obstacles, the end terminal (eP) and the position and direction of the initial TZM. In E^* the shorter TZPs cause either self-intersections or collisions, as shown in Fig.2.

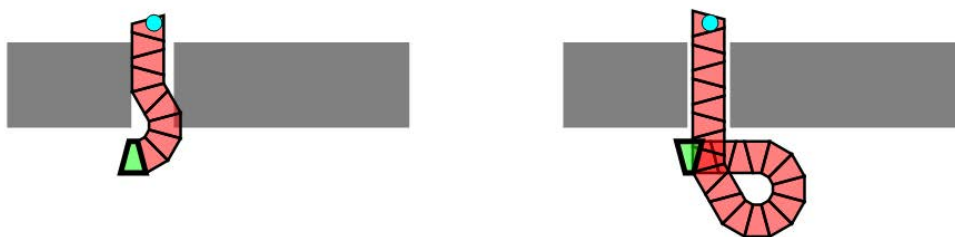


Figure 2: Two possible TZPs in the environment E^* . Two obstacles, initial TZM and the end terminal (eP) are shown in: gray, cyan and green, respectively. From left to right: both TZPs are non-allowable as they violate either: the obstacles, or self-intersection prohibition, respectively.

As Fig.2 indicates, the criterion reflecting the self-intersection prohibition must be embedded in the algorithm in order to generate allowable TZPs and ultimately – the optimal TZP.

4 Meta-heuristic approach: Evolution Strategy

In two-dimensional case, each trapezoidal unit can be placed in two alternative orientations (0 or 1). Therefore the number of all possible solutions grows exponentially, namely as 2^n , where n is the number of units. For example, there are 1,125,899,906,842,624 possible solutions for a 50-unit TZP. If the evaluation of each TZP took one millisecond, the entire process would require almost 36 millenia. Thus it is rational to use a population-based meta-heuristic method. In this paper it is the classic Evolution Strategy, which is based on intensive mutation and does not employ recombination.

4.1 Encoding of TZP in base-36 numbers

Although TZ structures are three-dimensional, in this paper, the problem is reduced to the projection on the 2D plane. A single trapezoid, called 1, which corresponds to units R and L_2 , and its rotation, called 0, which corresponds to L and R_2 , allow creating a path of virtually any trajectory. For a corresponding interactive demonstration, see [4]. The encoding of a TZP is straightforward. It is a binary string of 0s and 1s, corresponding to: the left- and right-turning trapezoids, respectively. The genetic operations are performed directly on such binary lists. This encoding can be “compressed” by simple conversion to the numbers of higher base. The higher is the numerical base, the shorter is the notation. Base-36 is the most compact case-insensitive alphanumeric numeral system which uses ASCII characters. In base-36 the digits are represented by the Latin letters $a \dots z$, and the Arabic numerals $0 \dots 9$. Since some TZPs have 0s in the beginning of the list of units which would be truncated in conversion to base-36 equivalents, a 1 is added at the beginning of each sequence. This compact encoding is particularly practical in comparisons among various TZPs, e.g.:

$$\{0, 0, 0, 1, 1\} \rightarrow \{1, 0, 0, 0, 1, 1\} \rightarrow 100011_2 \rightarrow z_{36}$$

4.2 Mutation types

Two types of mutation have been defined:

- “Flip mutation” changes the value at randomly selected loci to the opposite values. Its probability is controlled by parameter m_F : $0 \geq m_F \geq 1$, where 0 and 1 return: no change at all and mirror reflection of entire TZP, respectively.
- “Length alteration” changes the length of a genotype. One of three actions is randomly selected:
 - Remove the last unit from the genotype.
 - Append a unit to the genotype. The value of appended unit is randomly selected from $\{0,1\}$.
 - “Do nothing”.

This type of mutation is controlled by probability m_A . The probabilities of removal and addition of a unit are equal. This means that for $m_A = 0$ and $m_A = 1$: the genotype will not be altered, and will certainly be either shortened or extended, respectively.

The following subsection shows that both types of mutation are necessary for the convergence of the algorithm.

4.3 Quasi-experiment

In this quasi-experiment the convergence of algorithm from a random population to a given genotype is examined.

- Populations size: 100.
- Genotypes in the population: the sequences of random lengths between 20 and 50 integers from set $[0,1]$.
- The target genotype is a sequence of random integers $[0,1]$ of length 40.
- Selection type: “roulette”.
- 200 iterations unless the ideal solution is found.
- The selection is based on Hamming Distance between the target genotype and each individual in the population. If the compared genotypes have different lengths, that difference is added to the dissimilarity. The lower the dissimilarity, the higher is the probability of selection for the next generation.

Fig. 3 shows two plots with only one type of mutation applied compared to a plot where both mutations are applied.

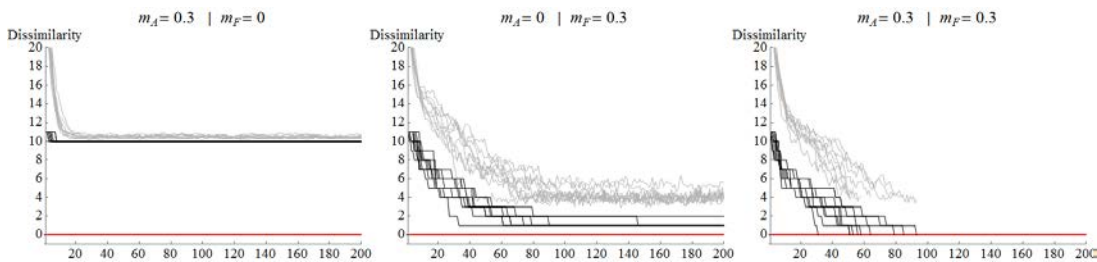


Figure 3: 10 trials of quasi-experiment in three set-ups. From the left: no “flip mutation” ($m_F = 0$), no “length alteration” mutation ($m_A = 0$), and both types of mutation combined. Red, black and gray indicate: 0 dissimilarity to the target genotype, the minimal and mean dissimilarities in each generation, respectively.

As Fig. 3 indicates, both types of mutation must be applied in order for the algorithm to converge.

4.4 The Cost Function

Proper formulation of the *objective function* is usually the most challenging issue in optimization. Here it measures three independent variables: the total number of units, the distance to the terminal eP and degree of collisions (both self-intersections and obstacle violations). Since all three are to be minimized, the objective function is called the *cost function*, CF for short. The number of units is obvious for each genotype since it is simply its length. The computation of the minimal distance to eP (called “reaching error” ϵ_r) and the collision violation (ϵ_c) are based on image processing of the rendered image of each individual TZP. The motivation for such decision was as follows:

1. According to preliminary experiments, this method seemed faster than combinatorial analysis of geometrical intersections.
2. Ultimately the same approach is to be applied to fully 3D Truss-Z optimization, with the use of *Mathematica*’s **Image3D** function.
3. The problem of quantitative measurement of intersections is even more discouragingly difficult with 3D solids than 2D convex polygons.

4.4.1 Collisions

The quantitative measurement of collision violation (ϵ_c) is based on the image processing of rendered potential solution (TZP) in the environment E^* . Fig.4 shows an example of a candidate solution.

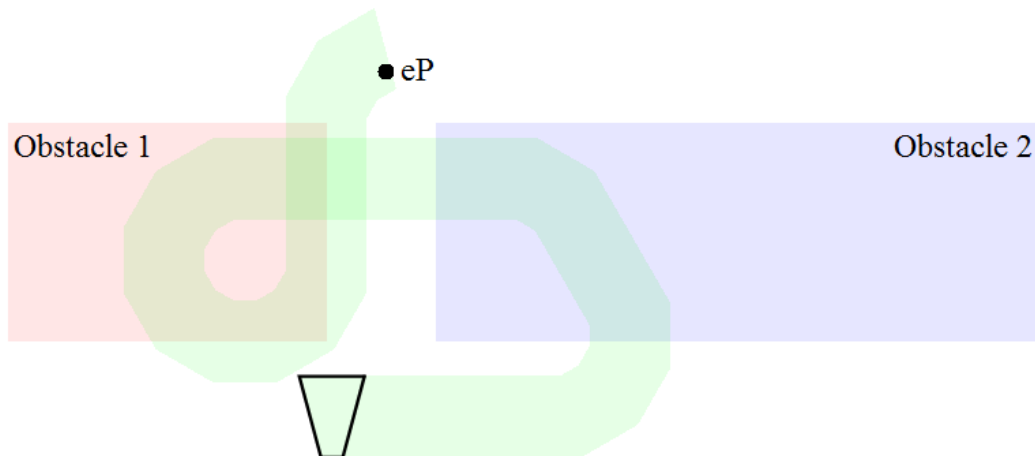


Figure 4: RGBA (red, green, blue and alpha) image of the environment E^* with two obstacles (shown in light red and blue), end terminal eP , initial unit outlined in black, and a sample TZP shown in light green. In other words, TZP uses green channel, and the obstacles use red and blue channels. Alpha channel is used of opacity.

As Fig.4 indicates, the objects are rendered with opacity, so the collisions can be easily detected and evaluated quantitatively. The collision violation (ϵ_c) is measured by the number of pixels of different colors. Fig.5 shows the collection of considered “collision colors”.


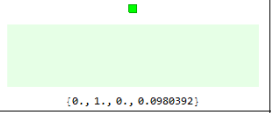
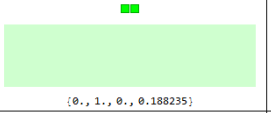
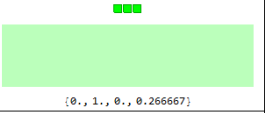
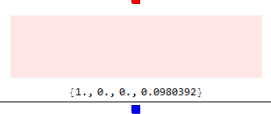
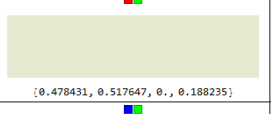
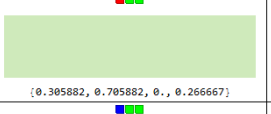
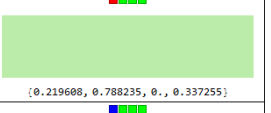
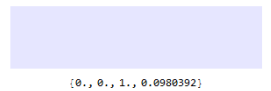
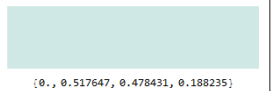
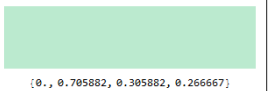
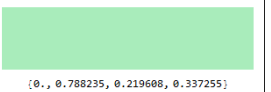
	 [0., 1., 0., 0.0980392]	 [0., 1., 0., 0.188235]	 [0., 1., 0., 0.266667]
 [1., 0., 0., 0.0980392]	 [0.478431, 0.517647, 0., 0.188235]	 [0.305882, 0.705882, 0., 0.266667]	 [0.219608, 0.788235, 0., 0.337255]
 [0., 0., 1., 0.0980392]	 [0., 0.517647, 0.478431, 0.188235]	 [0., 0.705882, 0.305882, 0.266667]	 [0., 0.788235, 0.219608, 0.337255]

Figure 5: RGBA colors used for the Cost Function CF . Each cell of this table contains the following: small sample of color/s in full opacity, the result of their overlapping and its RGBA value.

Detailed and quantitative information about objects overlapping can be derived from simple counting of the pixels of certain values. E.g. pixels of RGBA value: $\{0, 1, 0, 0.188235\}$ indicate areas where TZP units overlap once. RGBA values: $\{0.478431, 0.517647, 0, 0.188235\}$ and $\{0, 0.517647, 0.478431, 0.188235\}$ indicate the areas where TZP collides with obstacles 1 and 2, respectively, etc.

4.4.2 “Reaching error”

The distance of TZP to eP is also measured by an image processing method. Firstly, an analog of potential distance field, (DF), is generated for E^* as shown in Fig.6.



Figure 6: The analog of DF created with radial gradient fill spanning from eT to the farthest corner of considered E^* . The TZP “mask” and eP are shown as: transparent white and white disk, respectively.

Secondly, a “mask” is created from the considered TZP. The smallest value of DF within that mask measures the distance to eP .

4.4.3 Aggregate Objective Function

Finally, all three measurements are combined to a single Aggregate Objective Function (AOF) as follows:

$$AOF = n + w_r \times \varepsilon_r + w_c \times \varepsilon_c \quad (1)$$

where n , ε_r , and ε_c are: the number of TZP units, the “reaching error” and “collision error”. ε_r has values from the range: $[0,1]$ and does not require normalization. ε_c is normalized to the reference non-self-intersecting TZM of the same number of units, so that $0 \geq \varepsilon_c \geq 1$. w_r and w_c are parametric weights corresponding to ε_r , and ε_c , respectively.

4.4.4 Calibration of weights for AOF

Firstly, a number of TZPs of various quality have been generated. Secondly, they have been ordered according to the author’s intuition to reflect their decreasing quality, as shown in Fig.7.

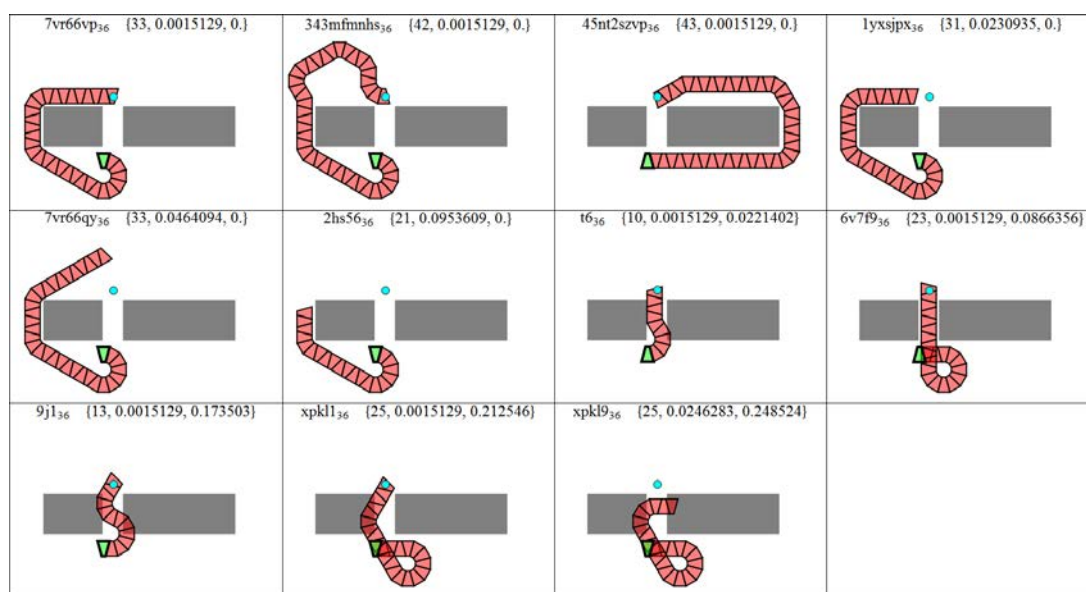


Figure 7: 11 TZPs ordered from the best to the worst according to the author’s judgment. On the top of each TZP its base-36 encoding and the list of $\{n, \varepsilon_r, \varepsilon_c\}$ are given.

The weights w_r and w_c have been “manually” adjusted, so that AOF is monotonically growing along with worsening quality of TZPs, as shown in Fig.8.

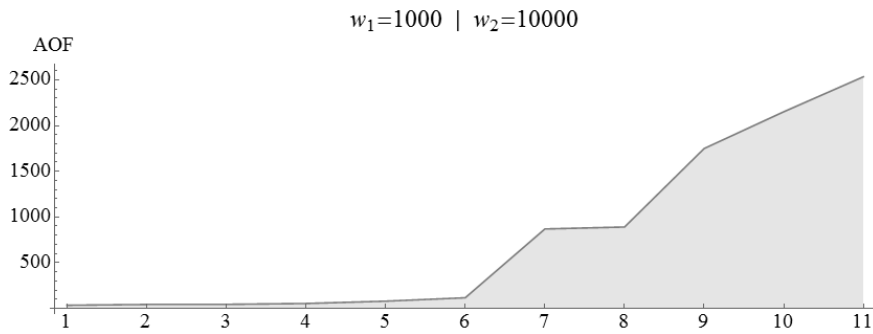


Figure 8: For $w_r = 1000$ and $w_c = 10000$ AOF grows approximately monotonically.

5 The experiment

The experiment has been carried out with various combinations of mutations' (flip & length alteration) probabilities, populations sizes, selection types, etc. The values of the parameters which produced repeatedly the most reasonable solutions are collected in Tab.1.

Table 1: The final set-up for the experiments

Population size	400
Selection type	Roulette
Mutations' probabilities	$m_A = 0.6, m_F = 0.3$
Maximum number of iterations	50

Fig. 9 shows a sample trial with the parameter set-up listed above and the best result with its base-36 encoding.

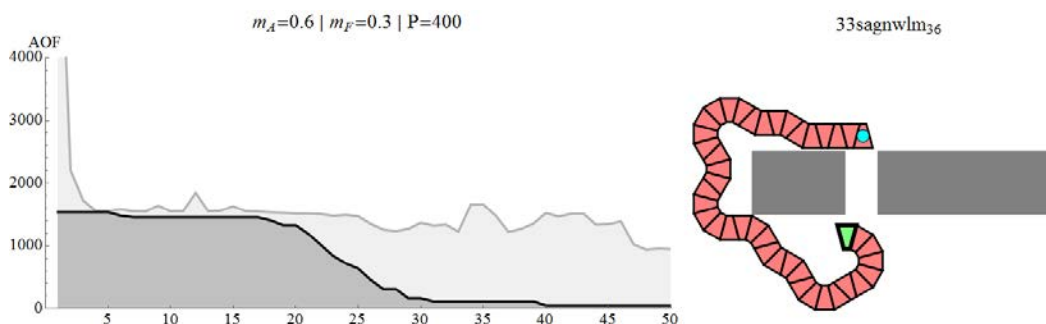


Figure 9: On the left: the convergence on the algorithm. Minimal and mean values in each generation are shown by black and gray lines, respectively. On the right: the best solution after 50 iterations.

6 Parallelization

The calculations described above take advantage of *Mathematica* functionality. Programming in this environment is very intuitive, however, the computation of the *CF* for a single TZP takes approximately 1 second on an Intel i7-3517U CPU @ 1.90 GHz 2.40 GHz notebook. As a result, a single generation of a population of size 400 takes approximately 6.5 minutes. Consequently, a 50-iteration experiment takes approximately 5.5 hours, which is far from satisfactory or comfortable.

Parallelization is the most natural direction, especially that the TZPs in every population form a table which can be easily split among several kernels. Our institution is equipped with a HPC cluster of four six-core Xeon X7460 CPUs @ 2.66 GHz with *Mathematica* installed. Theoretically, approximately a 24-times acceleration could be expected, which would result in less than 14 minutes experimentation time.

Unfortunately, in this case it is not possible. The graphics rendering functions used for *CF* calculation require *Mathematica*'s front end (i.e. graphical user interface, GUI for short). In other words, this process can not run on a headless server. Usually, due to licensing, the number of front-ends is much lower than for kernels. Therefore, in order to make use of several computers with *Mathematica* front-ends *Wolfram Lightweight Grid* must be used.

6.1 Wolfram Lightweight Grid

The *Wolfram Lightweight Grid (LWG)* is a system for launching, managing, and using *Mathematica* kernels across a network. It is particularly useful for creating an ad hoc grid built out of a collection of different types of computers. Potential benefits of *LWG* have been examined with a local network of just two computers: a notebook with MS Windows 8.1 and a desktop PC with MS Windows 7. The *LWG* server has been installed on the former one. Several combinations of kernel usage has been examined. The results have been collected in Table 2.

Table 2: Wolfram Lightweight Grid (*LWG*)

Cores →	1	2	4	6	8
Notebook i7-3517U CPU @ 1.90 GHz 2.40 GHz	6.59	4.8	3.7 (2 × 2 logical)		
Desktop i7-4790 CPU @ 4 × 3.60 GHz		2.68 (1+1)	1.79 (2+2)	1.14 (2+4)	1.22 (4 + 4)
	3.5	1.76	1 (4 × physical)		

As Table 2 indicates, in this particular problem, there was no clear benefit of using *LWG*, and the best performance was reached by the most straightforward setup on a single more powerful machine.

7 Cost function computation by GPGPU

There are other alternative methods of calculating the cost function (CF). In many cases the overlapping convex regions could be calculated quantitatively by geometrical methods, which are incomparably faster.

However, presented here image processing approach has certain benefits which are practically impossible to implement with other methods:

1. The obstacles can have absolutely free-form shapes including non-convex polygons, B-spline regions etc.
2. The penalty for violating such obstacles does not have to be constant.

As demonstrated above, *Mathematica* functions are very versatile but also quite time-consuming. However, high performance general-purpose computing on graphics processing units (GPGPUs) seem to be perfectly suited for such tasks. A simple program has been implemented in CUDA platform for testing. It evaluates all individuals in each population in two steps:

1. The entire population is rendered as a single image (texture) with the use of OpenGL.
2. The CF is calculated for all TZPs on that image with the use of CUDA.

Obviously, the size of this texture depends on the required size of sub-image for each TZP, and their number in population. E.g. assuming that each TZP requires an image representation of 600×600 pixels, and there are 256 of them, the resulting texture will have 9600×9600 pixels.

Since the obstacles are rectangular and TZP units are trapezoidal, all the population (including obstacles) are rendered with the use of quadrilaterals (quads). It is worth mentioning, that OpenGL is highly optimized and uses the graphics processing units (GPUs) very efficiently. As a result, such procedures are practically instantaneous. Indeed, as our calculations indicate, for a population larger than ≈ 100 , drawing a single TZP takes approximately 0.002 ms. For smaller populations the CPU-GPU memory transfer factor increases this time, however not significantly.

In the subsequent step, appropriate CUDA kernels are deployed to calculate all the overlapping regions in order to compute the CF . This can be executed in various ways, and the efficiency of the solution will depend on the reduction mechanism. In this paper a straightforward approach is analyzed, where a single CUDA block with threads calculates the overlapping regions for each individual TZP. Therefore the number of threads is equal to the size of the population, and its size is limited according to a particular device and CUDA specifications. Normally the maximum number of threads per block is 1024.

Figure 10 compares the dependence of the time T_{TZP} required to calculate the CF for a single TZP to the size of population (P) between CUDA and a serial program running on CPU.

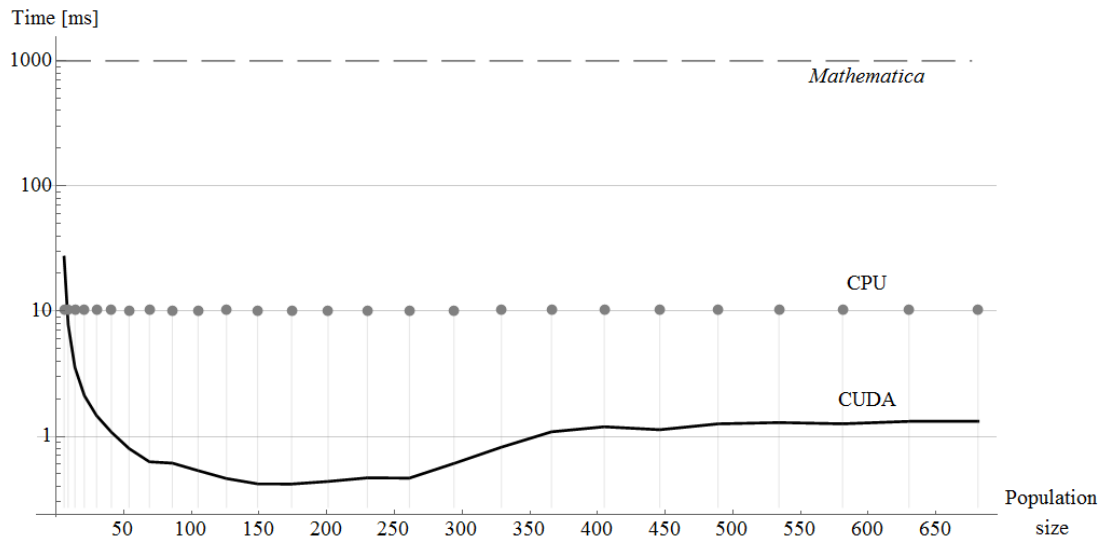


Figure 10: The log-plot showing dependence of CF calculation time (T_{TZP}) for an individual TZP in the population of size P . Hardware used: i7-4790 @ 3.6 GHz with GeForce GTX 960. The P number of CUDA threads and serial CPU process are shown by black line and gray dots, respectively. In both cases, the entire population is drawn with OpenGL and thus the parallel capabilities of the GPU are employed. The corresponding computation times in *Mathematica* is indicated by a dashed line.

As Fig.10 indicates, for small P , the computation time for T_{TZP} by CUDA decreases due to the overhead time for copying data between the host and the GPGPU device being shared among the TZPS. T_{TZP} is minimal at the point where the GPGPU can assign a single thread to all TZPs. Afterwards, the overhead related to switching threads in the block increases, and finally from $P \approx 400$, the scaling of T_{TZP} becomes approximately linear. Further optimization of the CUDA process can be done relatively easily. Nevertheless, presented here, simple solution speeds-up the CF computations substantially. As a result, a population of 500 TZPs can be evolved for 100 generations in approximately one minute.

8 Conclusion

A new, image processing-based optimization method of Truss-Z planar path (TZP) in an environment constrained by obstacles has been presented. A novel method of quantitative evaluation of TZP self-intersections and TZP obstacle violation is introduced. The optimization is based on a classic meta-heuristics, i.e. Evolution Strategy. The cost function (CF) used for evaluation of TZP is computationally expensive. Two parallelization methods are presented, namely: *Wolfram Lightweight Grid*, which was not particularly effective, and CUDA platform, which sped-up the optimization algorithm by a few ranges of magnitude.

Acknowledgments

This work was completed as part of the project titled: “Innovative Extremely Modular Systems for temporary and permanent deployable structures and habitats: development, modeling, evaluation & optimization”. It was funded by “Polonez 2” research grant no. 2016/21/P/ST8/03856 supported by the National Science Centre, Poland. This project has received funding from the European Union’s Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 665778.



References

- [1] M. Zawidzki, “Discrete Optimization in Architecture – Extremely Modular Systems”, SpringerBriefs in Architectural Design and Technology, Springer Singapore, 2016, ISBN 978-981-10-1109-2 (eBook), ISBN 978-981-10-1108-5 (soft-cover).
- [2] M. Zawidzki and K. Nishinari “Modular Pipe-Z System for Three-Dimensional Knots”, Journal for Geometry and Graphics, 17(1), 081-087, 2013.
- [3] M. Zawidzki, “Creating organic 3-dimensional structures for pedestrian traffic with reconfigurable modular Truss-Z system”, International Journal of Design & Nature and Ecodynamics 8(1), 61-87, 2013.
- [4] M. Zawidzki, “Tiling a path with a single trapezoid along a given curve”, Wolfram Demonstrations Project [published: 04.07.10].
- [5] M. Zawidzki, K. Nishinari, “Modular Truss-Z system for self-supporting skeletal free-form pedestrian networks”, Advances in Engineering Software, 47(1), 147-159, 2012.
- [6] M. Zawidzki, “Retrofitting of pedestrian overpass by Truss-Z modular systems using graph-theory approach”, Advances in Engineering Software, 81, 41-49, 2015.
- [7] M. Zawidzki, K. Nishinari, “Application of evolutionary algorithms for optimum layout of Truss-Z linkage in an environment with obstacles”, Advances in Engineering Software, 65, 43-59, 2013.
- [8] M. Zawidzki, T. Nagakura, “Foldable Truss-Z module”, Proceedings for ICGG 2014: 16th International Conference on Geometry and Graphics, Innsbruck, Austria, 4-8/8/2014.
- [9] M. Pratt, A. Geisow, “Surface/Surface Intersection Problems”. In J. Gregory (ed.): The Mathematics of Surfaces. Oxford University Press, 117142, 1986.