Effective Multi-objective Discrete Optimization of Truss-Z Layouts Using a GPU

Machi Zawidzki* and Jacek Szklarski

Department of Intelligent Systems,
Institute of Fundamental Technological Research,
Polish Academy of Sciences, Warsaw, Poland

Dated: September 2018

Abstract

Truss-Z (TZ) is an *Extremely Modular System* for creating skeletal free-form ramps and ramp networks. The TZ structures are comprised of four variations of two types of basic unit subjected to rotation. The two types of units are: R and L being a mirror reflection of each other. This paper presents a novel method based on image processing, evolutionary algorithm and intensive parallelization of multi-objective optimization of TZ layouts.

The algorithm returns a sequence of modules. The result guarantees a TZ connection between two given points (regions) and minimizes the fitness function representing certain costs associated with setting up the TZ structure.

The fitness function depends on the cost of TZ structure as well as the variety of costs related to the environment where the it is to be placed. E.g.: the earthworks, vegetation removal, obstacles avoidance, etc. There are no restrictions on the fitness function definition. It can depend on any variable which can be represented by a two-dimensional map of any property of the environment.

The formulation of the presented method is suited for application of well-established image processing methods which efficiently evaluate candidate solutions on a GPU. As a result, the employed genetic algorithm efficiently probes the search space. The practical applicability of this approach is demonstrated with three case-studies:

1. simultaneous paving of a path with congruent units in a hilly environment with trees & bushes and finding the best location for a pier over an existing river;
2. constructing of a TZ connector spanning over a mountain valley with lakes (where supports can not be placed);
3. retrofitting of an existing railway station with a large wheelchair TZ ramp of over 10 m elevation while preserving trees and minimizing the earthworks.

Keywords: Truss-Z, Extremely Modular System, retrofitting, accessibility, multi-objective, discrete, combinatorial, optimization, genetic algorithm, parallel computing, GPU, GPGPU.

1 Introduction

The idea of Extremely Modular System (EMS) has been introduced in [1]. In a nutshell, the purpose of EMS is to create free-form modular objects in given environments without obstacle violations, and self-collisions.

EMS jointly meets the following three criteria:

*Electronic address: zawidzki@mit.edu; Corresponding author

http://rcin.org.pl

1. EMS allows for creation of structurally sound free-form structures.
2. The number of module types in EMS is minimal.
3. EMS is not derived from a regular tessellation of space.

The fundamental property of EMS for design is that the EMS modules can be assembled in more than one way. The concepts of two EMSs: Truss-Z (TZ) and Pipe-Z (PZ) have been presented in References [2] and [3], respectively. Any two Pipe-Z modules can be connected at practically any relative twist angle, and any two Truss-Z modules (TZMs) in a single-branch structure can be connected in four distinct configurations.

1.1 The potential applications of EMS

Practically, every week brings events such as: natural, humanitarian and man-made disasters, in particular terrorist attacks, mass migration, etc. During such unexpected situations, the time is the most precious, and services provided by deployable and reconfigurable structures are invaluable. Rapidly deployable structures can save lives, health and improve human comfort in several ways:

- in case of disasters they can serve as escape routes, also for
- creating infrastructure:
  - for temporary use such as connector sleeves which can also serve to allocate persons according to the condition, contamination, religion, sex, age, formal status, etc.
  - for semi-permanent use such as refuge shelters, infirmary, etc.
- In case of settlements and habitats in extreme situations such as military conflicts, or extreme environments such as undersea, arctic, or outer-space outposts. In such cases, due to the harsh environment the assembly must be as simple, quick and robust ("error-proof") as possible.

Prefabrication and modularity are common ways of minimizing the construction cost. However, they also substantially limit the diversity of possible shapes of structures. There is a number of modular systems, where high modularity results in trivial, repetitive forms. The approach of EMS is different: here any shape, i.e. simple or complex, are non-trivial assemblies of not-oversimplified modules. The underlying principle inspired by Nature is that no shape or direction are preferred, e.g.: a straight line is only one of many possibilities of linking two points in space, not "the best" by default.

1.2 The new contributions of this paper

This work is a continuation of preliminary results presented in the conference paper [4], which introduced the problem of self-intersection to the optimization of a two-dimensional single-branch TZ path (TZP) of given length in constrained environment (E). The problem has been formulated there as follows: create a TZP from a start (P_0) to end point (P_1) without self-intersections and collisions in a simple E constrained by only two rectangular obstacles. Three independent objectives have been subjected to minimization: the total number of modules (n), the "reaching error" to P_1 and the "overlapping error". The criteria have been weighted and aggregated to the objective function u. The algorithm was based on a classic meta-heuristic – Evolution Strategy (ES) The new contributions of this paper:

- The introduction of the formal cost matrix $\mathbf{C}$. It allows for optimization of a TZP in environment E where obstacles can have any shape and the associated cost depending on location in E is expressed by a real number. As a result, the full advantage of well-established image processing tools can be used.

http://rcin.org.pl

- The introduction of the cost flow matrix **F**. It substantially improves the efficiency of the algorithm, so the good solutions can be found consistently and relatively quickly also in highly constrained environments.
- A typical genetic algorithm (GA) is applied, with three operators: mutation, crossover, and tournament selection.
- As opposed to the previous approach, here the length of the entire TZP (i.e., the number of required units) is unknown beforehand. The TZP length becomes a new variable which is also encoded in a genotype and subjected to optimization.
- An efficient implementation utilizing GPU is introduced, where hundreds of individuals are drawn onto a single image in card's memory and evaluated in parallel.
- In principle, the presented framework can handle any number of TZPs in one environment.
- Three distinct and relatively realistic case-studies are presented, discussed and visualized.

All this makes the introduced here approach feasible for real-life applications which is demonstrated with the case-studies discussed below. In principle, the presented new methods are significantly more reliable and accurate. It is due to evaluation of substantially more individuals and therefore probing of the search space more efficiently while providing highly flexible definition of the fitness function. The task of finding the optimal order of TZ modules is not trivial since small, local changes in the sequence forming TZP can significantly alter the global properties and profoundly affect the TZ structure evaluation. Therefore a TZP must be optimized globally.

A real-life, universal, skeletal TZ module is also a result of structural optimization which assures that it meets safety requirements under expected loads. This, however, is out of scope of this paper.

1.3 Truss-Z

Conceptually, 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. In a real-life general case, both units: R and L need to be physically fabricated. 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 Figure1.

Previous research on TZ system focused on discrete topological and geometric optimization of TZM [5]. Evolutionary algorithms have been successfully applied for optimization of single- and multi-branch TZ layouts, and the results have been presented in: [6] and [7], respectively. Exhaustive method, namely graph-theoretic depth-first search algorithm has been successfully applied for finding *the ideal* TZ single-branch layout in a highly constrained environment [8]. Moreover, primary analysis of deployable TZM has been presented in [9]. The issues of *structural design* of TZM including topological optimization of the placement of diagonals, the sizing of members' cross-sections, the selection of materials, assembly method, etc. have been addressed in [10–12].

All EMS structures, including TZ must be installed in a given environment without self- and obstacle violations. In general, the problem of geometrical detection of self-intersection is non-trivial [13], and for multiple-unit TZs becomes very hard. 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 modules would not self-intersect.

http://rcin.org.pl

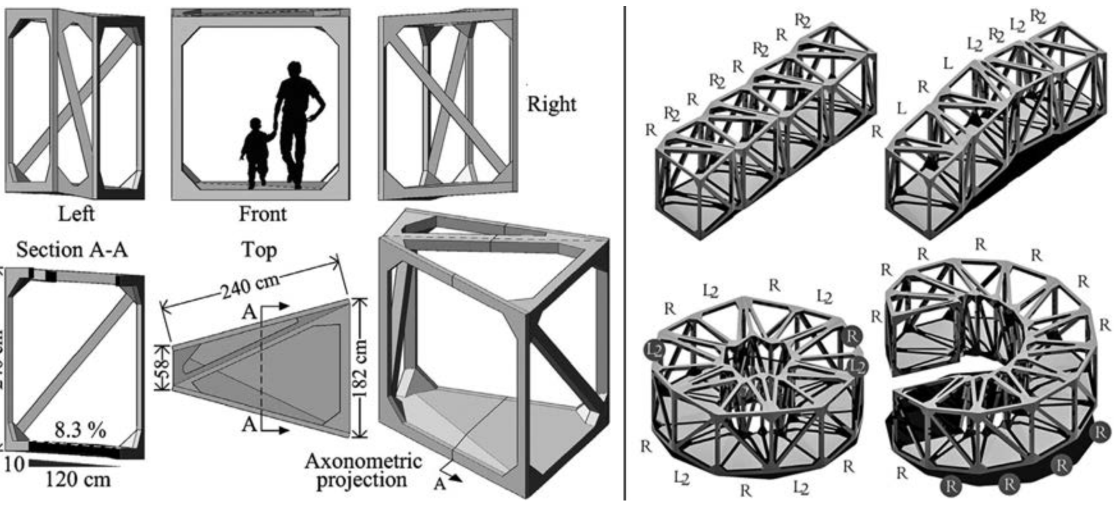

Figure 1: On the left: TZ basic unit (R). Upper row: orthographic views. Lower row: section A-A showing the slope, top view and axonometric view. On the right: some basic examples of single-branch TZ structures. Upper row: "straight and flat" with 8 units, "straight up & down" (8). Lower row: a flat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

1.4 Motivation and outline of the method

The motivation for this work is to take maximal advantage of the available "inexpensive" computer resources in order to successfully tackle more realistic challenges for installing TZs. This means:

- Development of a new model capable of handling more diverse environments
  - with various topographies
  - and types of obstacles of any form & cost
- and an And efficient implementation of an algorithm capable of handling such challenges.

As mentioned above, the method presented here combines: image processing, meta-heuristic and parallelization. For meta-heuristic, a classic evolutionary algorithm is appropriated. It has been successfully applied in the related fields of structural topology and shape optimization [14].

The nature of evolutionary algorithms makes them perfectly suitable for parallelization, including massive parallelism. Apart from classic approach based on distribution of a task among multiple CPUs, of particular importance are solutions based on programmable gate arrays (FPGA) or GPUs. Such algorithms can be accelerated in various ways, depending on specific problem and available hardware [15–19]. Usually, the most efficient solution is focused on speeding up calculation of the fitness function [20]. For a recent review on distributed evolutionary algorithms see [21].

In the presented work, a GPU is employed to calculate the objective function in parallel for each individual in the population.

In principle, this is similar to the approach described in [22] where GPU has been applied for continuous topological optimization in highly customized architectural design. However, in the approach presented here the structure is strictly modular and the nature of optimization is discrete (combinatorial).

The exact number of genotypes which can be evaluated depends on the hardware used. For example, in the method described below with the assumed parameters, a single graphical card with 4GB memory can calculate the objective function of ≈ 700 individuals for a single OpenGL and CUDA program call. The entire process requires less then a second. If a population size is larger,

http://rcin.org.pl

it needs to be divided into "chunks" of size suitable for the card's memory. There is an additional overhead due to data copying between the graphical device and the host. Due to availability of these fitness function massive calculations, the entire optimization procedure is $10^2 - 10^3$ times faster than with a single CPU. The exact value of speed-up depends on the following factors: population size, genotype length, size of considered environment for TZP, etc.

2 Problem statement

In principle, all Extremely Modular Systems are three-dimensional, so are the TZ structures. However, in this paper it is assumed that the optimization problem can be reduced to 2D. The TZ modules are projected on the 2D plane which gives two different trapezoids. The trapezoid denoted as '1', corresponds to modules R & L_2, and its rotation, denoted as '0', corresponds to L & R_2.

The computation objective is to find a sequence of $S = \{s_1, \ldots, s_l\}$, where $s_i \in \{0, 1\}$. The sequence S should connect point P_0 with P_1, meeting given optimization criteria. The sequence length l, i.e. the number of required trapezoids is unknown. Formally, it is assumed that the environment in which the TZP will be placed, can be represented by a two-dimensional $n \times m$ grid. Each element (i, j) of the grid is associated with a certain costs of installing the TZ structure. These costs are expressed as a real numbers $C_{i,j}$, and form the cost matrix $\mathbf{C}$. The exact, physical meaning of $C_{i,j}$ depends on the specific conditions of a considered case. For example, if an element (i, j) corresponds to a non-removable obstacle, then $C_{i,j}$ would be infinite.

In order to tackle the problem more efficiently, the cost flow matrix $\mathbf{F}$ is introduced. To find $\mathbf{F}$, first a grid-like graph G is created, where each node corresponds to the matrix element $C_{i,j}$. The node is connected to eight (or less for $i = 0, j = n$) neighbors: $C_{i-1,j-1}, C_{i,j-1}, C_{i+1,j-1}, C_{i,j-1}$, etc. Each edge is weighted, so the edge $E_{(i,j)-(k,l)} = 0.5(C_{i,j} + C_{k,l}) + c$. c is the constant minimal cost of TZP installation (e.g. for the "easiest possible" and homogeneous terrain in a single grid element). A cost $C_{i,j}$ is assigned infinity if the structure must not pass through a certain space. In such cases edges connecting to (i, j) are not created. The matrix representing the graph G can become quite large, with $(n \times m)^2$ elements. However, it is sparse and this property should be exploited in a real implementation. Having the graph G, Dijkstra algorithm using Fibonacci heaps is applied in order to find the total minimum cost $F_{i,j}$ leading from any point (i, j) to the final point P_1.

With every trapezoidal segment i at a specific location, there is an associated cost q_i. It is calculated as follows: a set O of indexes (k, l) is created so that any grid element (k, l) which overlaps (even partially) with the segment i is placed into O. Then,

$$q_i = \sum_{(m,n) \in O} \eta_1 F_{m,n} + \eta_2 C_{m,n}, \tag{1}$$

and the total cost for the sequence is:

$$Q_S = \sum_{i=1,\ldots,l} q_i, \tag{2}$$

η_1, η_2 being the parameters.

Without compromising the generality, it can be assumed that the first element of S is an L-trapezoid, whose middle point lies at the point P_0, and the base is parallel to the ordinate. The most straightforward, "greedy" approach to construct S would be to add the consecutive trapezoids so that each new one would lead to smaller increase of Q_S. However, this would prevent TZP from growing "uphill", i.e. against the gradient of $\mathbf{F}$ or $\mathbf{C}$. An example where such an approach would fail is discussed in Subsection. 2.3.

In order to overcome this issue, a certain kind of a backtracking algorithm could be implemented. However, considering that the cost matrix can be arbitrarily complex, and that the

http://rcin.org.pl

solution can be naturally encoded in a binary sequence, it is reasonable to tackle the problem with a typical genetic algorithm.

2.1 The implementation of GA

The binary coding is applied since it is the most natural representation of the problem in the applied genetic algorithm. Each individual in the genetic population is encoded with N_b bits, where the first n_l bits define the sequence length l, which is at least $l_{\min} + 1$. Consequently, the encoded structure can have variable length. Only the l bits located after the first n_l bits are used to define the TZ structure which is eligible for evaluation (the remaining last bits are ignored).

E.g. for $N_b = 16$, $n_l = 4$, $l_{\min} = 2$, the vector

$$0, 1, 1, 0, \ | \ 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0$$

encodes the sequence of length $l = l_{\min} + 0110_2 + 1 = 9$ (the vertical bar shown above denotes separation of the part encoding the sequence S length and the sequence itself). The total length is increased by 1 since the first segment L is fixed. Therefore, the above code yields the final sequence: $LRLRLRLRL$. In this case the last three bits of the genotype string are simply ignored. Note that in a population, all individuals are encoded using the same N_b bits. This means that usually, if variable length is allowed, for $n_l > 0$ there will be a number of unused bits at the end of such string representation.

The entire population consists of N_{pop} individuals which initially are random binary sequences of length N_b. The classic genetic operators are employed as follows:

- Mutation is applied to an individual with the probability $P_{\text{mut}} = 0.2$, where each bit is assigned a new random value with the probability $P_{\text{mset}} = 0.1$.
- Crossover between two individuals is performed at a random crossing point with probability $P_{\text{cx}} = 0.2$.
- Tournament selection is performed for $N_{\text{tour}} = 3$ genotypes. That is N_{tour} random individuals are drawn, and the best one is introduced into the subsequent population.

After the sensitivity study of the algorithm the probabilities for GA operators, selection method, number of bits for length encoding, etc. have been chosen. A simple problem with known ideal solution (Sec. 2.3) and a number of more complex examples have been studied at various set-ups. The convergence rates, population mean fitnesses, their standard deviations, etc. have been analyzed. The probabilities have been selected to maintain proper levels of exploration and exploitation of the search space.

The GA has been implemented with the use of DEAP library [23]. The calculation of the objective function was realized by a specially designed program which runs on a GPU.

2.2 Calculation of the objective function

The objective function $u(S)$ depends on: the sequence S, the constant matrix $\mathbf{C}$ (and therefore $\mathbf{F}$), and the points P_0, P_1. This requires calculating q_i for each segment, see Eq. (1). If the matrices $\mathbf{F}$ and $\mathbf{C}$ had certain well-defined, regular elements, it could be possible to find the value of q_i by analytical formulas. However, since the cost matrix can have any arbitrary values, each element $C_{i,j}$ overlapping (even partially) with the segment i must be considered, and its contribution to q_i must be also taken into account.

Moreover, since there are additional optimization criteria, the value of $u(S)$ is not simply equal to the cost function. There is a penalty for self-intersections $\alpha n_{\text{intrsct}}$, n_{intrsct} being the number of elements of the cost matrix which overlap with more than one trapezoid. Another penalty is imposed if a trapezoid overlaps an existing structure, i.e. where $C_{i,j}$ is infinite, βn_{obst}. Furthermore, there is a “reward” for getting closer to the destination point P_1, γd_{dest}, where d_{dest}

http://rcin.org.pl

is the smallest Euclidean distance between any element of TZP to P_1. This handling of a multi-objective optimization is straightforward. However, there are certain drawbacks, namely: the parameters α, β, and γ must be selected by a designer. The general form of the fitness function, which is subjected to minimization has the following form:

$$u(S) = Q_S + \alpha n_{\mathrm{intrsct}} + \beta n_{\mathrm{obst}} - \gamma d_{\mathrm{dest}}. \tag{3}$$

The final form of the fitness function, however, is different for each particular optimization case and depends on desired architectural qualities required by a designer. The above function is subjected to minimization, and Eq. (3) represents the total costs associated with the cost matrices $\mathbf{C}$, $\mathbf{F}$. It is the result of summation of Eq. (2), the penalty for self-intersection (proportional to α), the penalty for collision with obstacles (β) minus a "reward" for getting closer to the destination point (γ). The choice of numerical values of α, β, γ depends on the definition of the cost matrix, geometrical properties of the terrain, and expected costs associated with removing certain objects, e.g., existing vegetation. In cases where such vegetation is "worth preserving" for the reasons that go beyond simple economy, the cost of its removal requires individual assessment.

In practice, for any realistic case, the designer:

- decides about the size of real area which corresponds to a single pixel;
- defines the cost units;
- estimates with real numbers the costs of: building the TZ structure, earthworks, obstacle removal, etc.;
- optionally, includes additional constraints in the definition of $u(S)$. For example, if self-intersections are absolutely not allowed, the value of α should be very large in comparison to Q_S. Consequently, self-intersecting individuals will be quickly removed from future generations in the GA.

For the calculations presented below, the parameters α, β, γ, and others, were adjusted individually in order to address the design objectives in particular case. The motivation for choosing those values is presented in each respective subsection along with the case description.

In practical implementation, regardless of the choice of the parameters for the fitness function, in principle, the calculation of $u(S)$ is closely related to image processing. Therefore, high performance general-purpose computation on a GPU is well-suited for this task. In order to obtain the solutions presented in this paper, a computer program has been implemented on CUDA platform to calculate $u(S)$. It evaluates all individuals in the population in two steps:

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

Obviously, the size of this texture depends on the required size of sub-image for each individual and their number in population. E.g. assuming that the cost matrix $\mathbf{C}$ has dimensions $n \times m = 600 \times 600$, each TZP requires an image representation of 600×600 pixels. If there were $N_{\mathrm{pop}} = 256$ individuals in the entire population, the resulting texture would have 9600×9600 pixels.

If the number of individuals was large and the entire population could not fit into the graphic card memory, the rendering process would be divided into smaller parts which would be evaluated independently. Note that as a result of the genetic operators, it is likely that in a new generation there will be some unchanged individuals since they could have avoided the mutation or crossover. Therefore, there is no need to re-evaluate the fitness function of such individuals. The number of new candidate solutions which need to be evaluated depends on the probabilities of the applied operators.

All TZMs are trapezoidal in shape, therefore the entire population is rendered with the use of quadrilaterals (quads). It is worth mentioning, that OpenGL is highly optimized and uses the GPU very efficiently. As a result, such procedures are practically instantaneous. Indeed,

http://rcin.org.pl

as the calculations indicate, for a population larger than ≈ 100, rendering a single TZP takes approximately 0.002 ms. For smaller populations the CPU-GPU memory transfer factor increases this time, however, insignificantly.

In the subsequent step, appropriate CUDA kernels are deployed to calculate all the overlapping regions in order to compute the $u(S)$. 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. 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. Usually the maximum number of threads per block is 1024.

Figure 2 compares the dependence of the time t_u required to calculate the $u(S)$ for *a single* TZP to the size of population N_{pop} between CUDA and a serial program running on CPU. In both cases: OpenGL-CPU and OpenGL-CUDA, the entire population is rendered with OpenGL and the parallel capabilities of the GPU are employed. This plot and all the other results described in this paper have been produced by i7-4790 @ 3.6 GHz CPU with GeForce GTX 960.

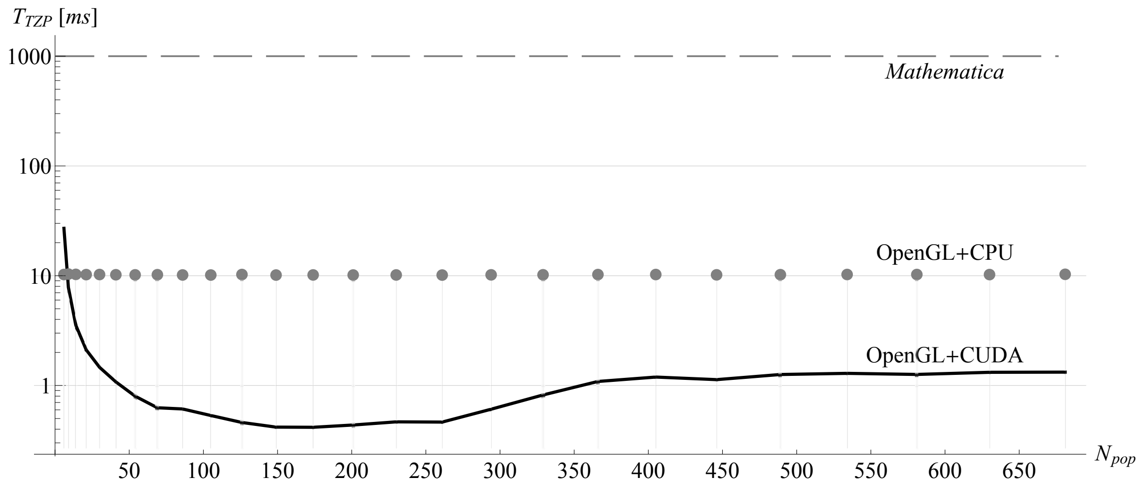

Figure 2: The log-plot showing dependence of the calculation time t_u of the fitness function $u(S)$ for a single genotype S, on the population size N_{pop}. The N_{pop} number of CUDA threads and serial CPU process are shown by: black line and gray dots, respectively. The corresponding computation times in high-level *Mathematica* uncompiled code is indicated by a dashed line.

As Figure 2 indicates, for small N_{pop}, the computation time t_u for $u(S)$ by CUDA decreases due to the overhead time for copying data between the host and the GPU device being “shared” among the individuals. t_u is minimal at the point where the GPU 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_u becomes approximately linear. Further optimization of the CUDA process can be done relatively easily. Nevertheless, the straightforward solution presented here substantially speeds-up the $u(S)$ computations. As a result, a population of size $N_{\mathrm{pop}} \approx 500$ can be evolved for 100 generations in approximately one minute (exact time depends on the assumed genotype length, mutation and crossover probabilities, etc.).

2.3 A basic example

In this subsection the problem of optimal TZP is illustrated with a simple example. Two scenarios are considered:

- TZP is to connect two given points (A and B) with a solid obstacle separating them;

http://rcin.org.pl

- As above, but additionally, there is a narrow passage in that obstacle, as shown in Figure 3.

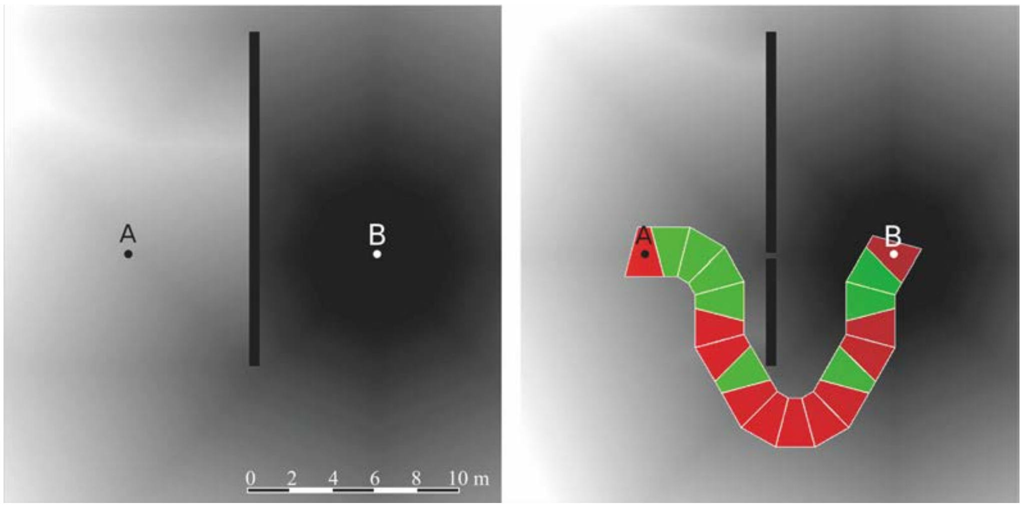

Figure 3: Gray-scale representation of two examples of the cost flow matrix **F** (black is 0, the brighter the color, the larger **F**). The black rectangles denote obstacles with infinite cost. A and B represent points: P_0 and P_1, respectively. The optimal TZP is shown on the right; L and R units are shown in red and green, respectively.

Without the narrow passage, in order to find the optimal solution, it is sufficient to build the TZP from the point A by adding consecutive trapezoids in a way which minimizes the fitness function. In other words, to follow the "downhill" of the gradient of **C** or **F**). However, if this passage is introduced, it modifies **F** in a way which makes such approach futile since following the gradient will result in a collision with the obstacle. Therefore, a more sophisticated search method is required, like the GA described above.

The TZP length is assumed to be $l_{\min} = 10$, and it is encoded with $N_b = 4$ bits, giving the maximum possible length equal to 26. The size of image is 300×300, the area of trapezoid is 450 pixels, the population size is $N_{\text{pop}} = 1024$ and all the individuals are rendered and evaluated in a single GPU step. Since this problem is relatively small, it is possible to evaluate all configurations by an exhaustive search method, so the ideal solution is known. This process takes approximately an hour. The weights for the fitness function Eq. (3) are: $\alpha = 5000, \beta = 1000, \gamma = 500$, $\eta_1 = 0.1, \eta_2 = 0$, the distance is measured in pixels (which applies for all the cases further in text). Figure 4 shows how the discussed GA converges to the global optimum (the value of objective function is divided by 10^5 for clarity). Notice that, on average, the total number of elements at first decreases and apparently the proper solution is build on shorter TZPs by adding new elements. Regarding the timing, the result is obtained in less than a second.

3 Three case-studies

In the following subsections, three relatively realistic case-studies are described. Although they are three-dimensional by nature, their problem formulation allows for successful implementation of the image processing framework introduced above which is purely two-dimensional. All three case-studies are defined as multi-objective optimization problems, namely:

1. **Modular Path**: Simultaneous paving of a path composed of trapezoids in a hilly environment with trees, and finding the best location for a pier over an existing river. Both the tree removal and the earthworks are to be minimized.

http://rcin.org.pl

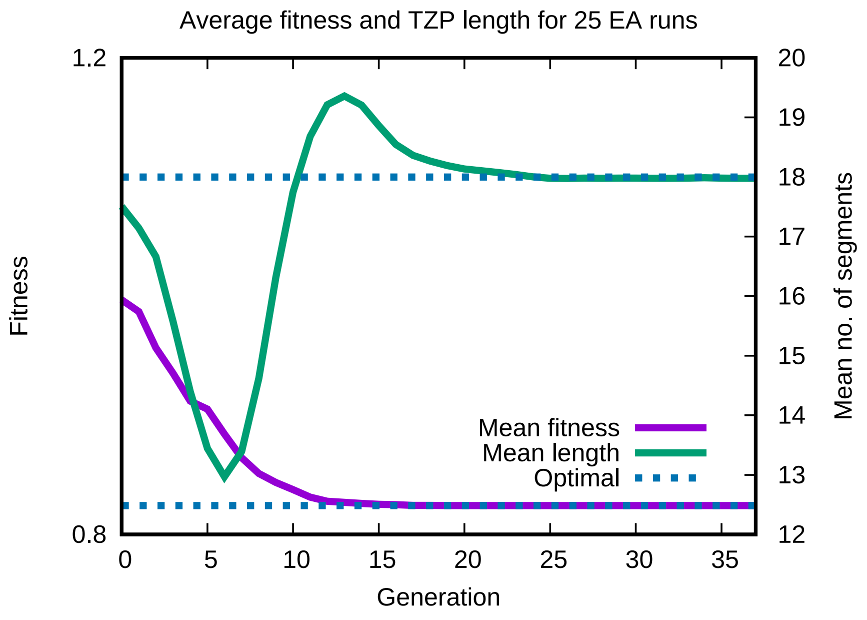

Figure 4: The average value of the fitness function and the average length of TZP in relation to the number of generations for 25 random initial populations. The globally optimal solution is found in approx. 15 steps.

2. **Mountain Pier**: Truss-Z flat connector spanning over a mountain valley with lakes.
3. **Railway station retrofitting**: Retrofitting of an existing railway station with a large semi-permanent wheelchair Truss-Z ramp of over 10 m elevation. There are several trees on site which ideally should be preserved. The earthworks also to be minimized.

Since these problems are relatively realistic and not extremely constrained, the number of possible solutions in each case becomes astronomical. E.g. in the last case, the number of all modules in TZ path is 101 (explained further in text). Thus a straightforward calculation gives 2^{101} of all potential solution which is absolutely unmanageable for exhaustive search methods. Therefore a classic heuristic method, namely genetic algorithm has been implemented in all three case-studies.

3.1 Modular path

In this case-study the following problem is considered:

- There is a hilly landscape with a river, trees, and bushes, as shown in Figure 5
- The values of: earthworks, tree & bush removal, and river crossing are set independently.
- Two given points: “start point” (P_0) and “end point” (P_1) to be connected by the modular path.
- The modular path to be comprised of congruent trapezoids which are projections of TZM on the horizontal plane. The maximal difference in elevation between each pair of such trapezoids should be less than 20 cm.

The cost matrix **C** is derived from the terrain height. This is rational since the expected paving should be as flat as possible, and any differences in elevation implicate a cost of earthworks.

http://rcin.org.pl

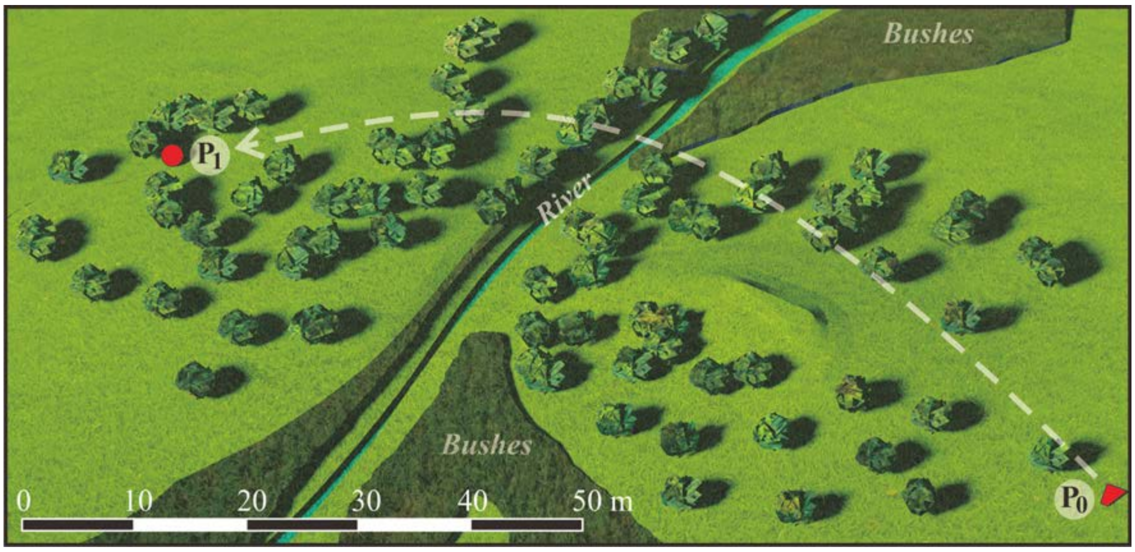

Figure 5: The axonometric visualization of the landscape where an optimal trapezoidal paving is to be installed. The location and orientation of the initial trapezoid are given at the start point P_0. The position of the endpoint P_1 is indicated by the red dot.

Additionally, **C** contains the cost of crossing the river, removing trees, or placing a TZP in the bushes.

It is difficult to assume the lengths of the optimal paving beforehand. Therefore a high variability is assumed: $l_{\text{min}} = 65, N_b = 7$. As a result, the possible lengths of pavings vary from 66 to 194. Image size for each individual is 720×512, $N_b = 1300$. The GA ran for 100 random initial populations for 500 generations. The objective function is a variation of Eq. (3) which takes into account the cost of removing trees (400 per pixel), the cost of placing a single trapezoid (1/3/6 per pixel for a regular/bush/river terrain, respectively), quadratic penalty for violating the assumed 20 cm elevation difference between the consecutive trapezoids, and the cost associated with a pier.

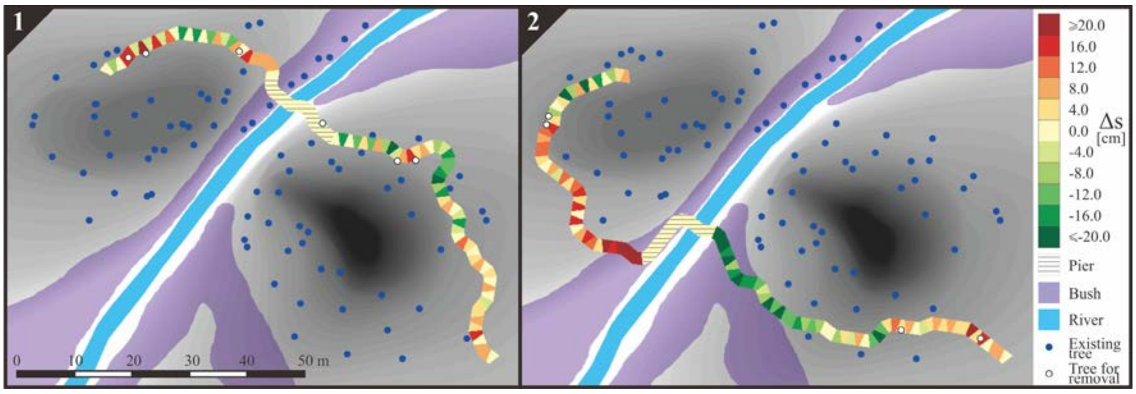

Figure 6: 1. The best result: 92 trapezoids, low earthworks and six trees to be removed. 2. A little worse alternative solution: 106 trapezoids, heavier earthworks and longer pier but less trees to be removed (only four).

At the river banks, the terrain becomes very steep. Therefore the condition of maximal step of 20 cm can not be met. In order to overcome this problem a *pier* is introduced. The pier is a

http://rcin.org.pl

sequence of trapezoids so that:

- The elevation difference of extreme points of the pier must be less than 40 cm. In other words, it is relatively flat.
- There can be only one pier.
- Pier is five times more expensive than a regular path. Its cost does not depend on the distance to the terrain below.
- The cost of pier depends on the surface type below. It is the least expensive to build pier over regular land, three times more expensive over bushes, and six times more expensive over the river.

Figure 6.1 shows the best result, and Figure 6.2 shows an alternative (second best) result. Both are superimposed on the cost matrix. The pier in the first solution is substantially shorter than in the second one. In the former – 12 units, in the latter – 22. Figure 7 shows the profiles of both solutions in more detail.

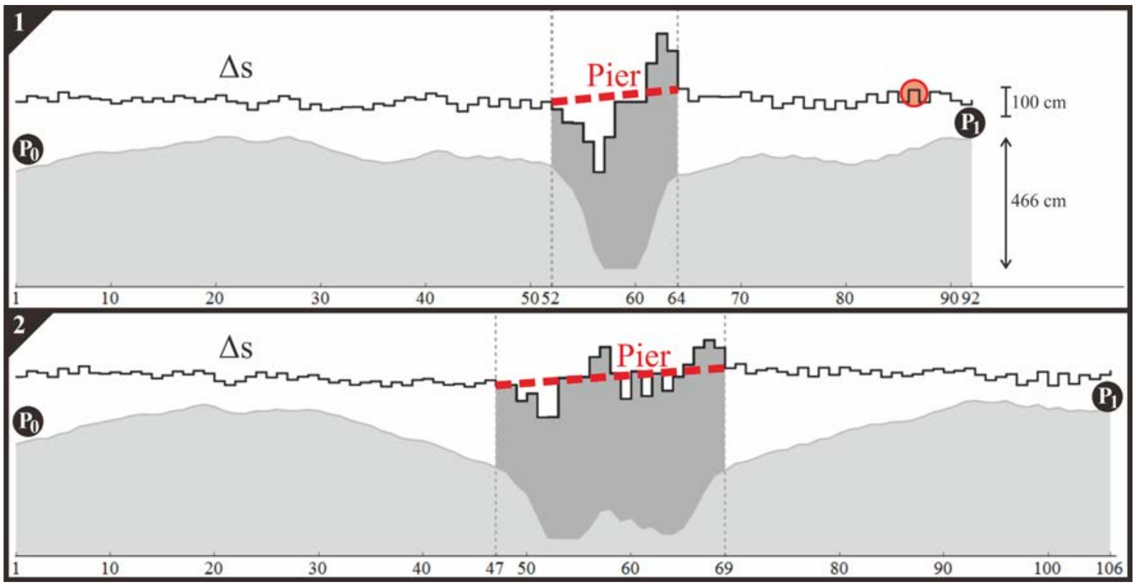

Figure 7: The profiles corresponding to the Figure 6. The terrain profile cut along the TZ paths is shown in light gray. The relative increase of subsequent risers is shown as black thick line above. This line can be considered as derivative of the slope of the terrain. The piers spanning over “illegal” trapezoids are indicated by red dotted line. Sub-figures 1 and 2: the profiles corresponding to the best and the second-best solutions.

As Figure 7.1 indicates, there is one trapezoid (indicated by red circle) which violates the 20 cm elevation limitation by approximately 2 cm. For comparison, Figure 7.2 shows the second-best solution where both the total length of the path and length of the pier are greater.

As mentioned above, the penalty for the riser hight violation is proportional to the square of distance above the assumed maximum (20 cm) squared. It is relatively severe ($\Delta h^2 \times 50$ per segment). Nevertheless, it might be more “economical” for the algorithm to violate this constraint in order to produce otherwise very good solution, as shown in Figure 7.1.

3.2 Mountain pier

Extreme environments such as outer-space habitats or high mountains here on Earth, are usually acquired by systems of bridgeheads. They are often in a form of self-contained stations which can

http://rcin.org.pl

be (at least partially) prefabricated and deployed in desired locations. Since the relative positions among such stations can not be fully predicted, creating connections among them may be difficult. In this case-study, the following scenario is considered: the environment is defined as a valley with three lakes between two mountain ridges, and the challenge is to create a flat Truss-Z pier spanning over the saddle and connecting two given points P_0 and P_1. The altitudes of P_0 and P_1 are equal. This problem is formulated as a constrained multi-objective optimization:

1. Minimize the number of TZ modules;
2. Minimize the cost (in other words, the total length) of supports, and
3. Avoid placing the supports in the lakes (forbidden zones).

The first criterion is straightforward as attaching every TZM has a certain cost. The third criterion is also intuitive, as placement of a support in a lake is much more expensive than on a solid ground and in some locations it is strictly forbidden. The second criterion is the cost of supports which is proportional to their total length. Here, the cost of support is linear and depends directly on the vertical distance from terrain to a respective TZM. In other words, placing TZMs along an isohypse reduces the cost of supports, but elongates the entire structure increasing its total cost. In general the placement of supports depends on assumed allowable maximal span of TZ. In this case-study three fixed spans are considered: 0 (i.e. a support is placed under each TZM), 5 and 10 modules. The position and orientation of the initial TZM at P_0 are set arbitrarily. Regarding the cost matrix $\mathbf{C}$, as in the previous example, it is derived from the height-map, taking into account areas where placing TZP is prohibited. Here, however, based on assumed spans, each fifth or tenth TZM enters the equation (1), and is included in the fitness function. Thus, for spans equal to five and ten, the resulting TZP can pass over areas of infinite cost (since they are simply ignored). Size of $\mathbf{C}$ is 450×450, $N_b = 1300$, 20 runs for 200 generations, the cost of placing of a pixel of support varies between 0 and 160 (depending on the terrain height), the support size is 45 pixels. Figure 8 shows the 3D visualizations of the results.

As Figure 8 indicates, the minimization of the total cost of the TZ structure is achieved by striking the balance between the cost of adding units and the cost of the supports. In the case of the supports placed under each TZM (see Figure 8.1) the TZ path runs between the lakes (forbidden zones) and after passing the valley runs along the second slope to minimize the total length of supports.

Sub-figures 8.2a and 8.2b show two alternative solutions with the supports placed under every fifth TZM. In the first case, the path is similar to the previous solution and is shorter by only one unit. In the second case, the length of the TZ structure is shorter by nine units and there are two supports less. However, since the total length of supports is substantially higher (see the part of TZ approaching P_1 in Figure 8.2b in particular), the overall cost of the entire structure is slightly higher (by approximately 1–2%). Also note how the largest lake partially "squeezes" here between the sixth and seventh support

Sub-figure 8.3 shows the most obvious case with relatively long spans of ten units. Here the TZ is almost straight from P_0 to P_1. Nevertheless, is seems like such long spans are not likely to be structurally achievable, thus spans closer to five seem more realistic.

http://rcin.org.pl

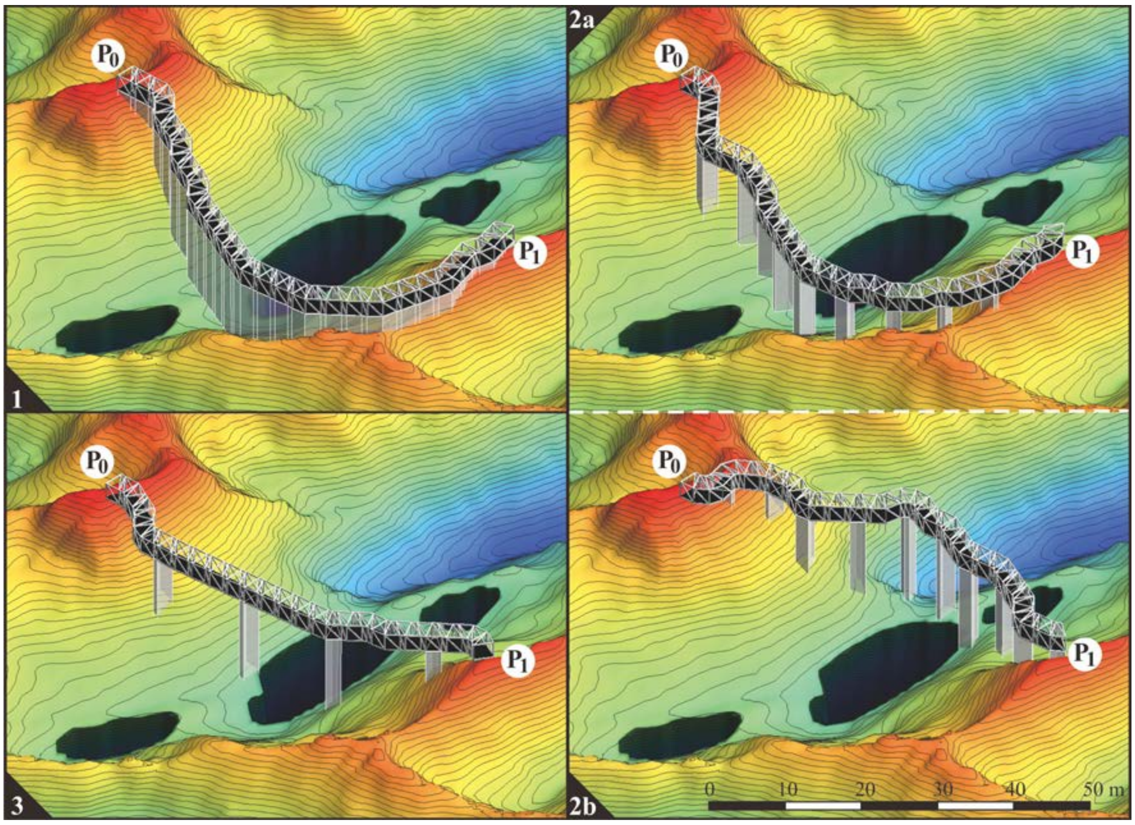

Figure 8: The axonometric visualization of the TZ pier at various spans of the supports. 1. The supports under each TZM; the number of TZMs is 60. 2a. The supports under every fifth TZM. The number of TZMs is 59. 2b. An alternative solution with nine fewer TZMs and two fewer supports. 3. Supports under every tenth TZM. The long span between supports allows TZ to almost "step over" the largest lake. The number of TZMs is substantially lower (45).

http://rcin.org.pl

3.3 Railway station retrofitting

Warszawa Powiśle is a railway station in Warsaw, Poland built in 1954–1962. Located in the district of Powiśle stretching between Jerozolimskie Avenue at the Charles de Gaulle roundabout (high-level entrance, E_H for short) and the intersection of 3 Maja Avenue and Kruczkowskiego Street (low-level entrance, E_L for short). Figure 9 shows the historical photographs soon after the completion and of the original architectural scale model.

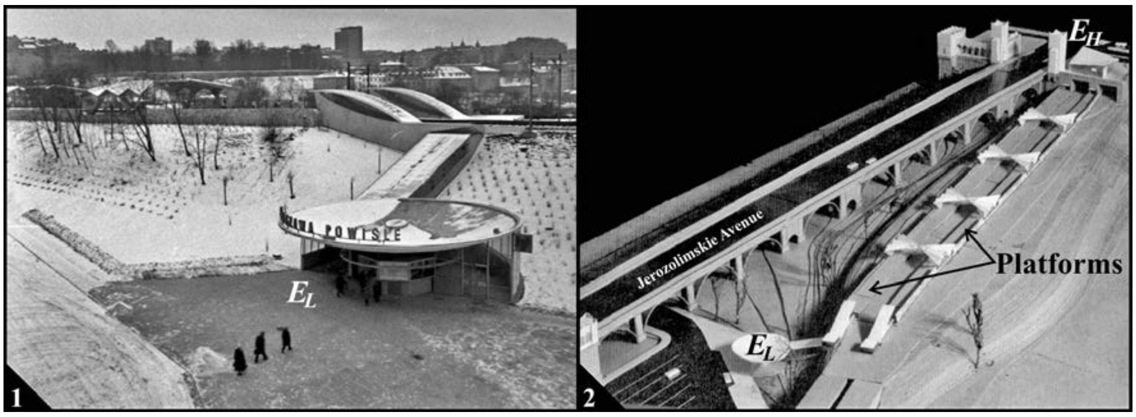

Figure 9: 1. The photograph taken from Jerozolimskie Avenue (1960s). 2. The original architectural scale model of the Warszawa Powiśle railway station (probably mid 1950s).

The station is located on a rail embankment extending from the Vistula river terrace (on which the city center is built) and ends at its western side with the entrance to the Cross-City tunnel. The access to the station is poor. The elevation difference between the street level and train platforms at the entrance E_H is over 5 m. This entrance is equipped with a standard stairway running down to the platforms. It is retrofitted with a stair-lift. However, presently the organization of its use is very problematic. The elevation difference at the entrance E_L is over 10 m. It is not equipped with any handicap facility. As a result, getting safely from E_L for a disabled person is practically impossible, as shown in Figure 10.1.

As Figure 10.1. indicates, the shortest distance between E_L and E_H is approximately 300m, as shown by the yellow line. However, a person on a wheelchair, in order to get from E_L to the platform, and without necessity of struggling with stairways, would have to travel preposterous 1.6 km, as shown by the red line.

Polish State Railways have vague plans for future major remodeling. The documentation published already in 2007 is available here [24]. However, meanwhile this station could be retrofitted with TZ which would serve as a semi-permanent inexpensive pedestrian ramp. Finally, as Figure 10.2 indicates, the south side of the rail embankment is presently covered with vegetation which can be considered valuable. Here the scenario and conditions are defined as follows:

- Create a TZ ramp providing direct access from the low-level entrance (E_L) to the closer (south) platform.
- P_0 becomes E_L. In this case, however, the endpoint of TZ is not strictly defined. Instead, it is an area adjacent to the platform as shown in Figure 10.
- The elevation difference between the ends of TZ is 10.1 m.
- The preliminary condition is that the TZ structure must be as short as possible. Therefore, since each TZM travels 10 cm upwards, the entire TZ must be comprised of exactly 101 units.
- The topography and the positions of existing trees are given.

http://rcin.org.pl

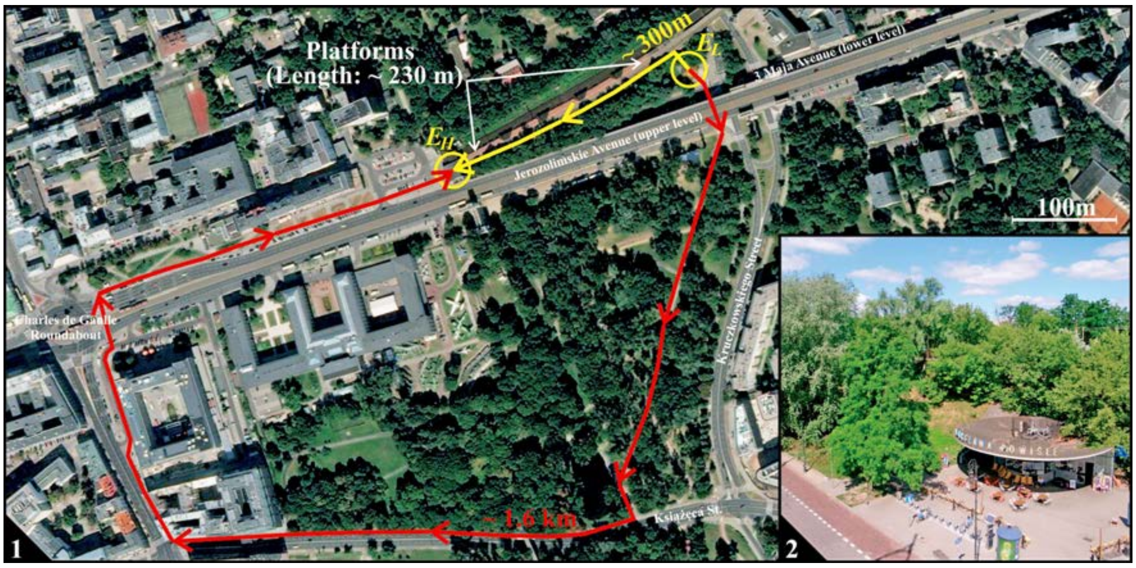
Platforms
(Length: ~230 m)
~300m
E_L
E_H
3 Maja Avenue (lower level)
Jerozolimskie Avenue (upper level)
100m
Charles de Gaulle Roundabout
Kruczkowskiego Street
Książęca St.
~1.6 km
1
2

Figure 10: 1. The aerial photograph showing the situation. The platforms of Warszawa Powiśle Railway Station approximately span between entrances E_H and E_L. 2. The photograph taken from Jerozolimskie Avenue towards entrance E_L.

- The optimization criteria: minimal earthworks, tree-preservation and *straightness*.

Figure 11 shows the design conditions for this case-study. The proposed TZ will provide direct access to the south platform of the station. Presently, the communication between both platforms at the east side of the station is realized via tunnel under the tracks. In this study, two additional regular ramps are proposed for improved accessibility, as shown in cyan in Figure 11.

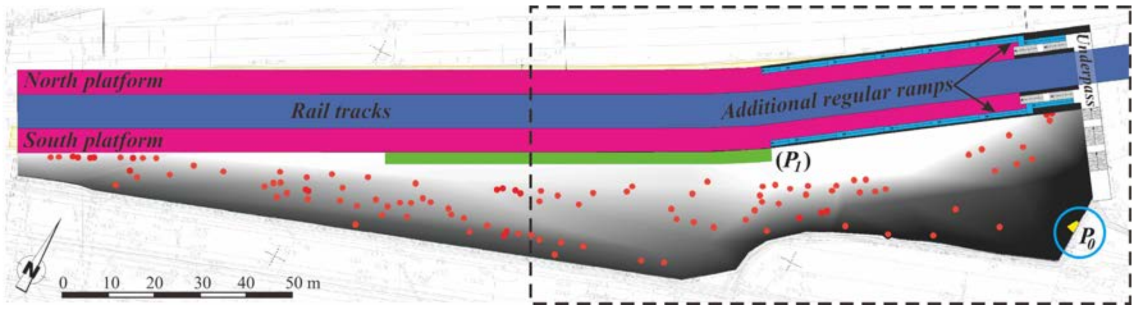

Figure 11: The design conditions. The railway track and platforms are shown in blue and magenta, respectively. Green indicates the target area for TZ (which corresponds to P_1). Existing trees are shown in red. The first TZM and its direction are set arbitrarily in P_0 and shown in yellow and circled in cyan. The *hight-map* is shown in gray-scale: white and black correspond to: the highest and lowest points, respectively. Dashed rectangle indicates the cropping area which will be shown in the subsequent figures.

In this case-study, there are three disjunctive aspects of TZ installation: the earthworks, tree preservation and "straightness", as described below.

http://rcin.org.pl

3.3.1 Earthworks minimization

This case is a "2.5D" problem. This means that the image processing is performed on regular images which by nature are flat. However, certain aspects of three-dimensionality are included in the algorithm. In particular, the earthworks are measured using information about the Z^{th} coordinates. The elevations of the (sampled by rasterization) points of the terrain are represented by the height-map, as shown in Figure 10. The calculation of the vertical position of a given trapezoid (being a projection of and representing a TZM) is straightforward. As mentioned above, the TZ will be comprised of exactly 101 units from elevation 0 to 10.1 m. This means that the mean altitude of a given trapezoid derives directly from its index in the sequence. This is reflected in the color of that trapezoid. E.g. values of blue channels of the initial and final trapezoids are: 0 and 100, respectively.

The height difference between the given trapezoid and terrain below is calculated by subtracting the pixel values of the centroid of that trapezoid and corresponding pixel of the height-map. If the result is positive – the trapezoid is above the terrain (i.e. adding earth is required), if negative – below (i.e. earth removal is required). In this example the cost of both actions i.e. adding and removing earth, is assumed to be equal.

All these requirements must be taken into account when calculating the fitness function. Since the TZP climbs uphill steadily with 0.1 m/segment, the cost of a segment depends on its particular elevation above the terrain (and therefore its location in the sequence). Consequently, the segment cost will be defined by the following variation of Eq. (1):

$$q_i = \sum_{(m,n)\in O} |h_{m,n} - i * 0.1|, \tag{4}$$

where $h_{m,n}$ represents the terrain height, and i is the segment number in the sequence.

This value enters Eq. (3) with appropriate weights related to penalty for overlapping a tree, enormous penalties for: self-intersection and entering forbidden region (outside the are designated for the ramp) or straightness. The size of $\mathbf{C}$ is 1200×200, $N_b = 1440$, each case ran for 30 random initial populations and 250 generations. The exact values used for calculating u depend on designer's preference. For example, sub-figure 12.1 shows the TZ which minimizes the earthworks. Here TZ almost exactly follows the existing slope at the expense of removal of nine trees.

3.3.2 Tree-preservation

In this set-up the penalty for tree-violation is very high (50 times higher than in the previous case). It is proportional to the number of pixels of a violated tree. A single tree is represented here by a circle of 1.5 m in diameter which corresponds to approximately 42 pixels. As a result, here the TZ meanders in order to avoid the trees as much as possible at he expense of the earthworks, as shown in sub-figure 12.2.

3.3.3 The optimal solution

In this case both earthworks and tree removal have been minimized according to authors' intuition, as shown in sub-figure 12.3. In this solution the earthworks are substantially less intensive than in the previous solution and only two trees are barely "shaved" (by just a few pixels). It means that it is likely that even these trees could be saved. Nevertheless, the earthworks are visibly more intensive than in the first solution.

3.3.4 The optimal-straight solution

In this case an additional criterion of "straightness" is introduced. It measures "how straight" is the run of a TZ and is equal to 2× number of substrings "0,1" in the sequence divided by its length. For example, a sequence of alternating opposite TZMs (e.g.: $0, 1, 0, 1, 0 \dots$) form a straight

http://rcin.org.pl

path with straightness = 1. Conversely, the straightness of a sequence of TZMs without any pair of alternating units (e.g. $0, 0, 0, 0 \dots$) is 0. Sub-figure 12.4 shows the solution which minimizes: the earthworks and tree removal and maximizes the straightness (which enters u multiplied by 10^5). In this case the amount of earthworks is slightly higher and two trees would be removed. On the other hand, the TZ structure is much more straight which can be a highly valuable functional feature.

This case-study is relatively realistic, thus finding a single ideal solution is very difficult if possible at all. The considered aspects of construction (earthworks, tree-preservation and straightness) have different dimensions. Since the combination of such criteria into a single objective function is not straightforward, the decision regarding their relative importance is difficult and rather arbitrary. As illustrated in Figure 12, it is up to the designer or investor to decide which solution is more suitable.

http://rcin.org.pl

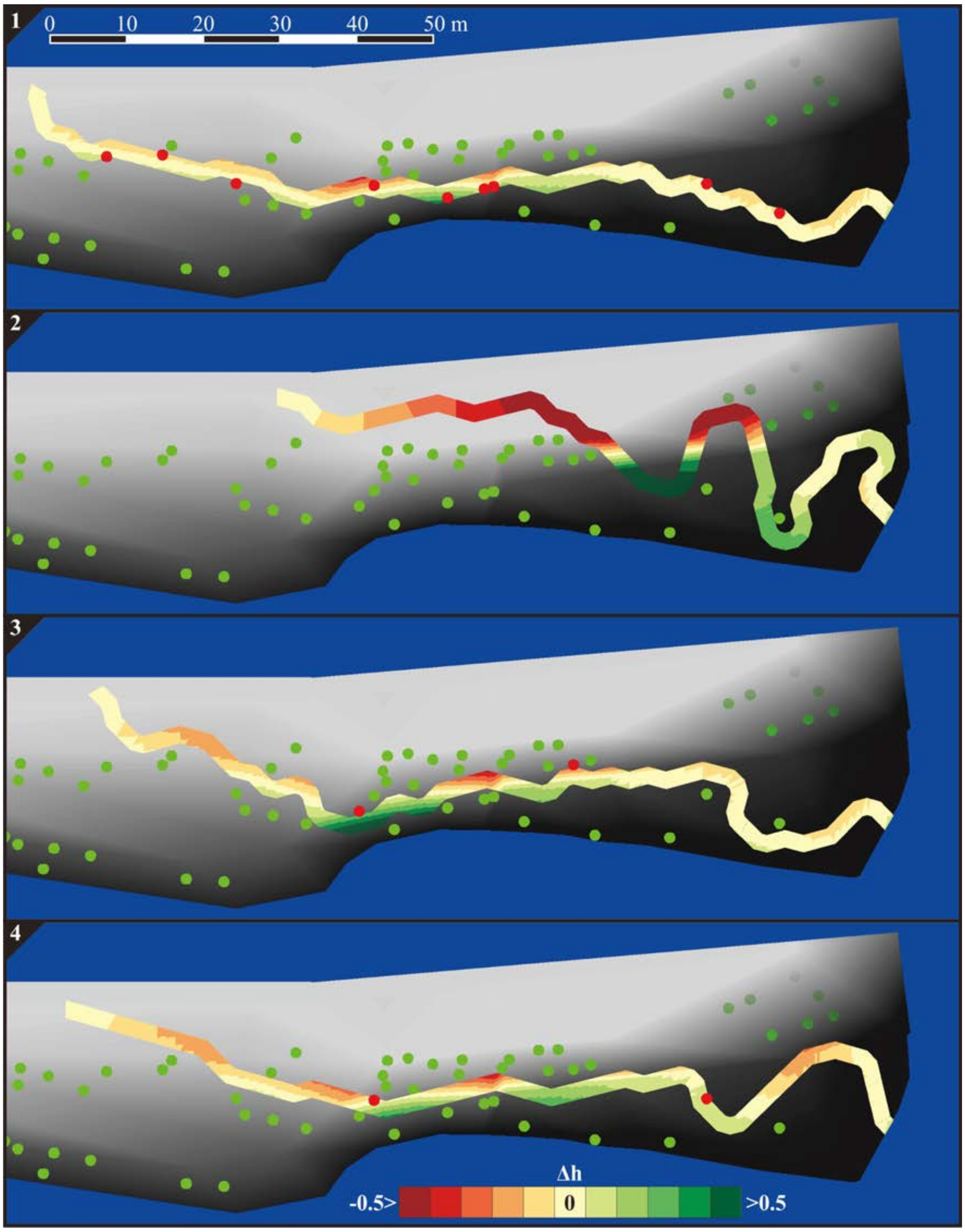

Figure 12: Four alternative solutions: 1. The minimization of the earthworks (at the expense of nine trees). 2. The preservation of the existing trees. 3. The optimal solution: earthworks and tree removal are minimal (two trees "shaved" but probably preservable). 4. The optimal-straight solution: earthworks and tree removal are minimal (removal of just two trees) and the straightness is maximized. The legend on the top reflects the degree of earthworks: extreme left and right colors correspond to: removal and built-up of over 0.5 m of earth, respectively, at 0.125 m increments. Middle color in the legend means that TZ follows exactly the slope. Dots indicate the existing trees; red and green indicate: trees to be removed and preserved, respectively.

http://rcin.org.pl

4 Conclusions and future work

A novel framework based on image processing, evolutionary algorithm and intensive parallelization has been introduced and applied to optimization of a modular system – Truss-Z. To the best knowledge of the authors such approach has not been applied to this kind of engineering problems yet.

The framework has been introduced and explained with a basic example. Next, three relatively realistic case-studies where TZ has been subjected to multi-objective optimization have been presented and analyzed:

- Simultaneous paving of a path composed of trapezoids in a hilly environment with trees & bushes and finding the best location for a pier over an existing river. Both the tree removal and the earthworks has been minimized.
- Constructing of a Truss-Z connector spanning over a mountain valley with lakes.
- Retrofitting of an existing railway station with a large semi-permanent wheelchair Truss-Z ramp of over 10 m elevation. There are several trees on site which ideally should be preserved, and the earthworks have been minimized.

For each case competitive solutions have been generated and discussed. The most important advantage of the presented framework is the use of a highly optimized GPU, which allows for effective generation and evaluation of large numbers of potential solutions. All computations presented here have been performed on a regular desktop PC. For each experiment, the objective function has been computed approximately $10^7 - 10^8$ times for the given number of initial populations (the entire process for each presented case took several hours). It is worth mentioning that the objective functions contained several image-based weighted variables and conditions including self-intersection and object-collision prohibitions.

Presently, the framework operates on 2.5D problems and addresses only the single-branch TZ structures. In the future research, it will be further developed for full three-dimensionality, and multi-branch TZ configurations. The former can be relatively easily implemented utilizing the existing, efficient techniques using z-buffering, the latter by appropriate candidate solution encoding in the GA. More realistic TZ retrofitting challenges will also be considered.

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 Unions Horizon 2020 research and innovation programme under the Marie Skłodowska–Curie grant agreement No 665778, .

References

[1] Machi Zawidzki. *Discrete Optimization in Architecture: Extremely Modular Systems.* Springer, 2017.

[2] M Zawidzki. Creating organic three-dimensional structures for pedestrian traffic with reconfigurable modular Truss-Z system. *International Journal of Design & Nature and Ecodynamics*, 8(1):61–87, 2013.

[3] Machi Zawidzki and Katsuhiro Nishinari. Modular Pipe-Z system for three-dimensional knots. *Journal for Geometry and Graphics*, 17(1):81–87, 2013.

http://rcin.org.pl

[4] M. Zawidzki and J. Szklarski. Single-branch Truss-Z optimization based on image processing and evolution strategy. In *Proceedings of the Fifth International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering*, pages 1–13, Stirlingshire, UK, 2017. Civil-Comp Press. ISBN 1759-3433.

[5] Machi Zawidzki and Katsuhiro Nishinari. Modular Truss-Z system for self-supporting skeletal free-form pedestrian networks. *Advances in Engineering Software*, 47(1):147–159, 2012.

[6] Machi Zawidzki and Katsuhiro 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.

[7] Machi Zawidzki. Optimization of multi-branch Truss-Z based on evolution strategy. *Advances in Engineering Software*, 100:113–125, 2016.

[8] Machi Zawidzki. Retrofitting of pedestrian overpass by Truss-Z modular systems using graph-theory approach. *Advances in Engineering Software*, 81:41–49, 2015.

[9] M Zawidzki and T Nagakura. Foldable Truss-Z module. *Proceedings for ICGG*, pages 4–8, 2014.

[10] Machi Zawidzki and Łukasz Jankowski. Multicriterial optimization of geometrical and structural properties of the basic module of a single-branch truss-z structure. In *World Congress of Structural and Multidisciplinary Optimisation*, pages 163–174. Springer, 2017.

[11] Machi Zawidzki, Łukasz Jankowski, and Jacek Szklarski. Structural optimization of a five-unit single-branch Truss-Z modular structure. In *Proceedings of the 8th ECCOMAS Thematic Conference on Smart Structures and Materials (SMaRT 2017), Madrid, Spain*, pages 5–8, 2017.

[12] Machi Zawidzki and Łukasz Jankowski. Optimization of modular Truss-Z by minimum-mass design under equivalent stress constraint. *Smart Structures & Systems.* (under review).

[13] Michael J Pratt and AD Geisow. Surface/surface intersection problems. *The mathematics of surfaces*, 6:117–142, 1986.

[14] David J. Munk, Gareth A. Vio, and Grant P. Steven. Topology and shape optimization methods using evolutionary algorithms: a review. *Structural and Multidisciplinary Optimization*, 52(3):613–631, Sep. 2015. ISSN 1615-1488.

[15] Wolfgang Banzhaf and Simon Harding. Accelerating evolutionary computation with graphics processing units. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, pages 3237–3286, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-505-5.

[16] R. Nowotniak and J. Kucharski. GPU-based massively parallel implementation of meta-heuristic algorithms. *Automatyka / Akademia Grniczo-Hutnicza im. Stanisawa Staszica w Krakowie*, T. 15, z. 3:595–611, 2011.

[17] N. Soca, J. L. Blengio, M. Pedemonte, and P. Ezzatti. PUGACE, a cellular evolutionary algorithm framework on GPUs. In *IEEE Congress on Evolutionary Computation*, pages 1–8, July 2010.

[18] Man Leung Wong. Parallel multi-objective evolutionary algorithms on graphics processing units. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, pages 2515–2522, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-505-5.

http://rcin.org.pl

[19] Tien-Tsin Wong and Man Leung Wong. *Parallel Evolutionary Algorithms on Consumer-Level Graphics Processing Unit*, pages 133–155. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006. ISBN 978-3-540-32839-1.

[20] Juan A. Gomez-Pulido, Miguel A. Vega-Rodriguez, Juan M. Sanchez-Perez, Silvio Priem-Mendes, and Vitor Carreira. Accelerating floating-point fitness functions in evolutionary algorithms: a FPGA-CPU-GPU performance comparison. *Genetic Programming and Evolvable Machines*, 12(4):403–427, 2011. ISSN 1573-7632.

[21] Yue-Jiao Gong, Wei-Neng Chen, Zhi-Hui Zhan, Jun Zhang, Yun Li, Qingfu Zhang, and Jing-Jing Li. Distributed evolutionary algorithms and their models: A survey of the state-of-the-art. *Applied Soft Computing*, 34:286–300, 2015. ISSN 1568-4946.

[22] J. Martínez-Frutos and D. Herrero-Pérez. *Massively Parallel Evolutionary Structural Optimization for High Resolution Architecture Design*, pages 29–49. Saxe-Coburg Publications, Stirlingshire, UK, 2017. doi: 10.4203/csets.40.3.

[23] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012.

[24] Robert Chwiałkowski. Kolej w Warszawie i województwie mazowieckim (Railway in Warsaw and Masovian Voivodeship), 2017. URL `http://siskom.waw.pl/kp-dworce.htm/powisle/`. Website.

http://rcin.org.pl

View publication stats