CAMES, **27**(2–3): 155–176, 2020, doi: 10.24423/cames.291

The Overview of Optimization Methods Applied to Truss-Z Modular System

Machi ZAWIDZKI

Institute of Fundamental Technological Research
Polish Academy of Sciences
Pawinskiego 5B, 02-106 Warsaw, Poland
*Corresponding Author e-mail: zawidzki@mit.edu

Extremely Modular Systems (EMSs) are comprised of as few types of modules as possible and allow creating structurally sound free-form structures that are not constrained by a regular tessellation of space. Truss-Z is the first EMS introduced, and its purpose is to create free-form pedestrian ramps and ramp networks in any given environment. This paper presents an overview of various multi-objective optimization methods applied to Truss-Z structures.

Keywords: Truss-Z, extremely modular system, discrete optimization, multi-objective.

1. Introduction

Mass prefabrication through modularity is a common method of minimizing the cost and time of construction [1]. However, modular systems usually drastically limit the diversity of the forms of possible structures. As a result, they are suboptimal. This is the price for the economization, which, however, should be mitigated by enhancing the design of such structures within the modularity constraints. Although the research in the field of structural optimization has been flourishing for decades, the optimization of modular structures is a relatively unexplored niche with only a few published results.

An EMS, introduced a few years ago in [2], is a novel approach to the design of architectural and engineering forms and structures. It is a family of concepts where assembly of congruent units allows for the creation of free-form shapes. EMSs are multidisciplinary and gradually gain attention in various fields of research: architecture, civil and space engineering, structural mechanics and computer science. There are four fundamental advantages of EMSs:

- economical – as they are suitable for mass fabrication, thus lowering the cost so they can be broadly applied;

- functional – as they allow for reconfiguration, expansion, reduction, rapid deployment;
- robustness – since every module that failed can be easily replaced with an identical but functional one;
- scientific – as they are suitable for intelligent mathematical modeling.

The disadvantage of EMSs, however, is that their assembly and control are non-intuitive and rather difficult. In other words, combining non-trivial congruent units to form meaningful structures or their kinematic actions are computationally expensive.

Truss-Z (TZ) is a modular system comprised of one truss-frame hybrid unit (and its mirror reflection), which allows to create ramps of free-form shape and constant slope, as shown in Fig. 1. This arrangement of modules has been produced by an algorithm based on graph theory (explained further in the article).

Fig. 1. Photograph of a physical scale model of retrofitting an existing overpass with 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. They can be assembled in two additional ways by rotation (R_2 – rotated R and L_2 – rotated L), effectively giving four types of units. The basic module and some examples ("straight and flat" with 8 modules, "straight up & down" (8 modules), a flat ring (12 modules), and a helix (12 modules) are shown in Fig. 2.

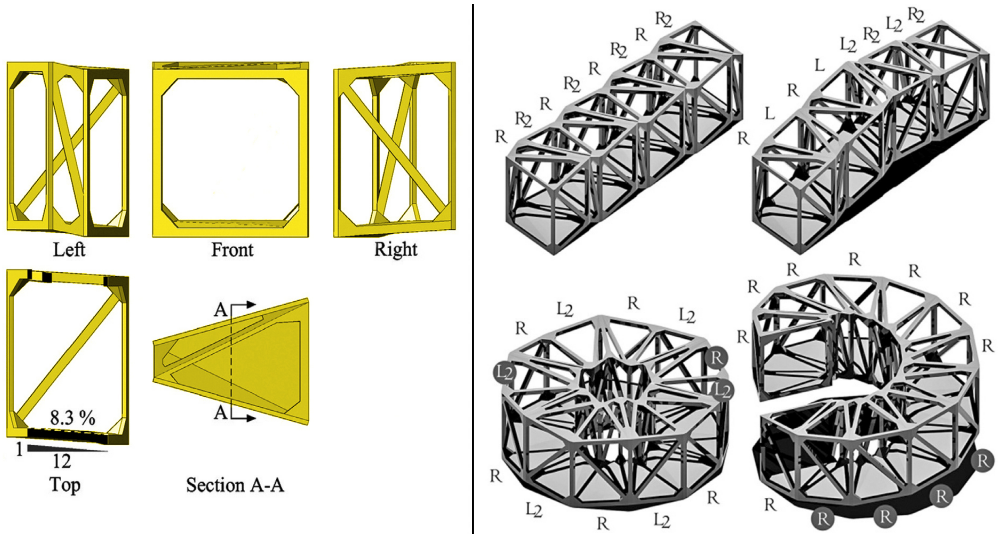

Fig. 2. On the left: TZ basic unit (R). Upper row: orthographic views. Lower row: section A–A showing the slope, top view and axonometric view. On the right: some basic examples of single-branch TZ structures.

This paper is an extended version of a presentation titled "Selected applications of computational intelligence methods to optimization in architecture" given at the Workshop on Engineering Optimization held in Warsaw, Poland on November 4th, 2019 [3]. It focuses on Truss-Z modular system and presents an overview of various computational methods applied to it in the past.

2. The optimization methods applied to Truss-Z

The first six methods described below have been applied in order to optimize the geometrical arrangement of TZ modules, and the last method presented pertains to the optimization of the internal structure of the TZ module.

2.1. Alignment to the given path

In this method, the guide path (GP) and the start point must be specified beforehand. The global optimization can be formulated as follows: 1) the Truss-Z path (TZP) to "adhere" to the GP, and 2) the TZP to be comprised of minimal number of modules. Figure 3 shows an example where the GP is given as a parametric curve. The number of units in this TZP is 33 (minimum). The list of units: {R L R_2 R R L R R R_2 R R L R R L L L_2 R L L L_2 L R L L L L_2 L R L L L_2 R}.

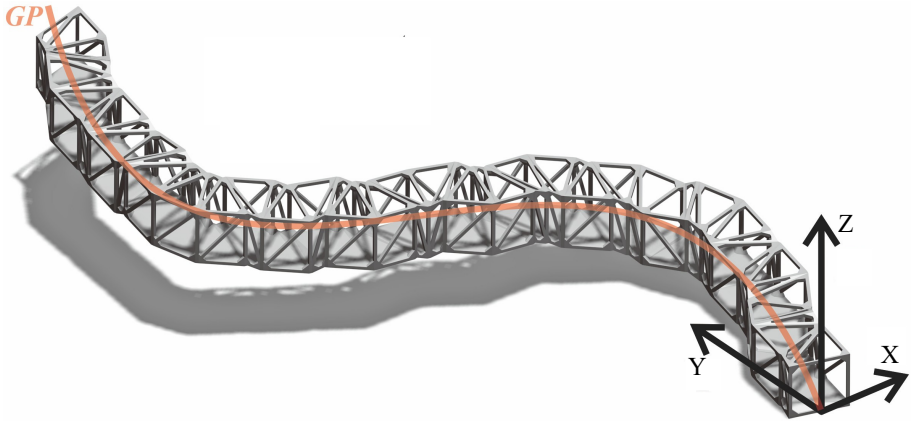

Fig. 3. A given parametric guide path and aligned TZ modules.

The sequence of TZ modules should follow the GP as "well as possible". Here this problem formulated as minimization is as follows:

$$\text{Minimize}\left(\frac{a}{b}d_i + (1-a)(1 - v_i \cdot r'[s])\right), \tag{1}$$

where d_i is the smallest distance between the centroid (C_i) of an ith unit and the point s on the curve r (guide path), v_i is the vector of an i-th unit, $r'[s]$ is the direction of the guide path r in the point s, a is the weight ranging from 0 to 1, which balances the influence from angle θ_i, and b adjusts the ratio between distance d_i and angle θ_i, which cannot be normalized.

At each step, the list of two values related to the given guide path is calculated for all four possible configurations where, in general, the weighted sum of distance d_i to the guide path r and angle θ_i are different for each case, as shown in Fig. 4.

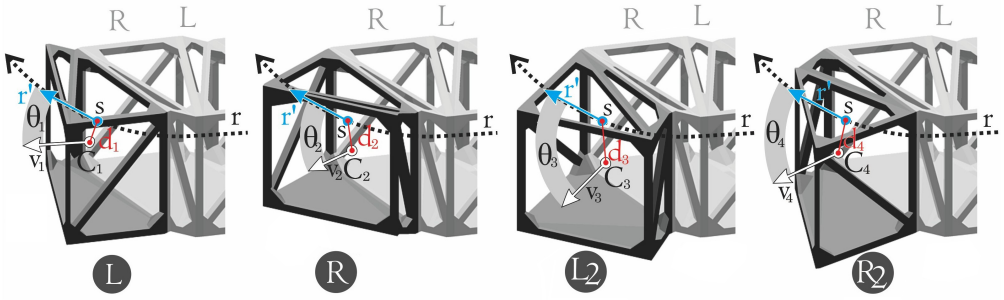

Fig. 4. Four possible configurations for the subsequent unit.
From the left: R, L (mirror reflection of R), L_2 (rotation of L) and R_2 (rotation of R).

The parameter a is a weight ranging from 0 to 1 and moves the influence from the normalized dot product of the GP's direction and the unit's vector

to the distance between the centroid of a unit and the GP. Since the objective function depends both on distance and angle, which cannot be normalized, the parameter b adjusts the ratio between them. Figures 5–7 illustrate the influence of these parameters for the same GP as in Fig. 3. In Fig. 5, $a = 0.5$, $b = 18$; this setting minimizes the distance from the centroid of every new unit to the GP. As a result, the TZP becomes "coarse" and in excessive "meandering" (the number of modules is 34).

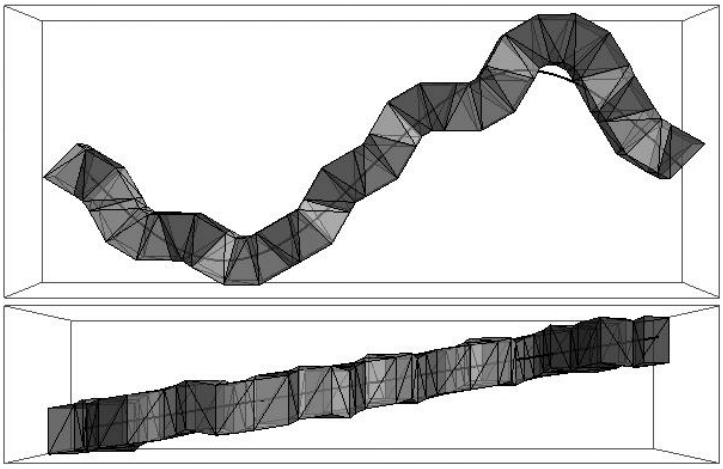

FIG. 5. Top and bottom sub-figures show the top and side perspective views, respectively. The GP is the same as in Fig. 3; $a = 0.5$, $b = 18$.

In Fig. 6, $a = 0$, thus, b becomes irrelevant; this TZP is "smooth" and follows the curvature of the GP. The number of modules is 29, however, this TZP does not reach the end of the GP, thus, it is not allowable.

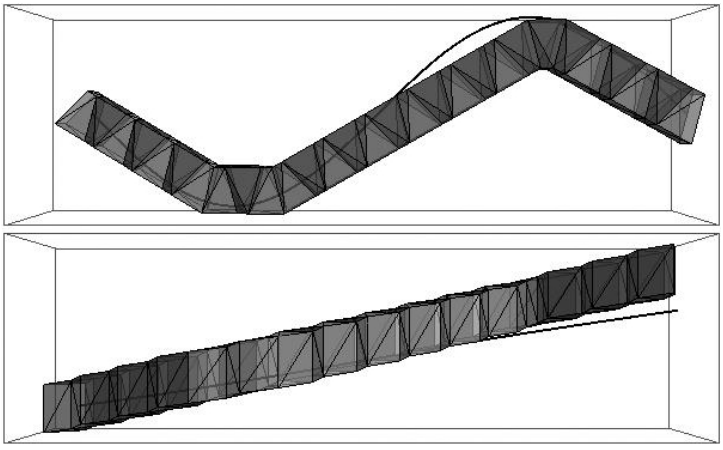

FIG. 6. A TZP for $a = 0$.

In Fig. 7, $a = 0.5$, $b = 100$; the balance is made between maintaining a small distance to the GP and following its curvature. The number of units to follow the given path is 33.

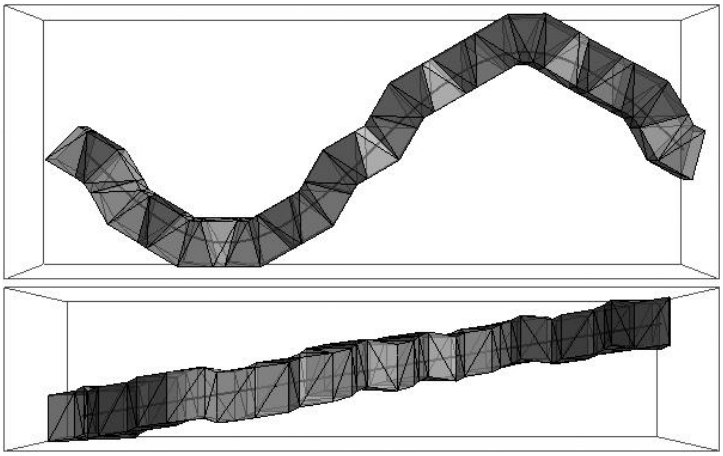

FIG. 7. A TZP for $a = 0.5$, $b = 100$.

The best alignment for this GP (shown in Figs 3 and 7) has been found by setting the parameters a and b by trial-and-error.

This procedure has been further developed for generating multi-branch structures. An example of creating a truss network for six terminals in an environment with three obstacles is shown in Fig. 8.1. The relationship between a potential unit U_i and the considered terminal t is shown in Fig. 8.2.

The procedure is based on constructing splines and then aligning the modules along them.

1) At first, two consecutive terminals are selected and the first spline guide path is created.
2) Secondly, the first run of the truss is created along the first path using the local optimization alignment formula (1).
3) Next, in order to connect the remaining four terminals (3, 4, 5 and 6) – so-called "junction units" (JU) are attached to the first run between the terminals 1 and 2.
4) A single junction unit can connect one (JU_1, JU_2) or two ($JU_3 = JU_4$) terminals. The location of junction units can be assigned directly or by a local discrete minimization using the following formula:

$$\text{Minimize}\left(\frac{a}{b}d_i + (1-a)\cdot(\mathbf{T}_i \cdot \mathbf{V}_i)\right), \tag{2}$$

where d_i is the distance between a centroid (C_i) of the i-th potential junction unit and a terminal t, $\mathbf{V}_i$ is the vector of a potential junction unit,

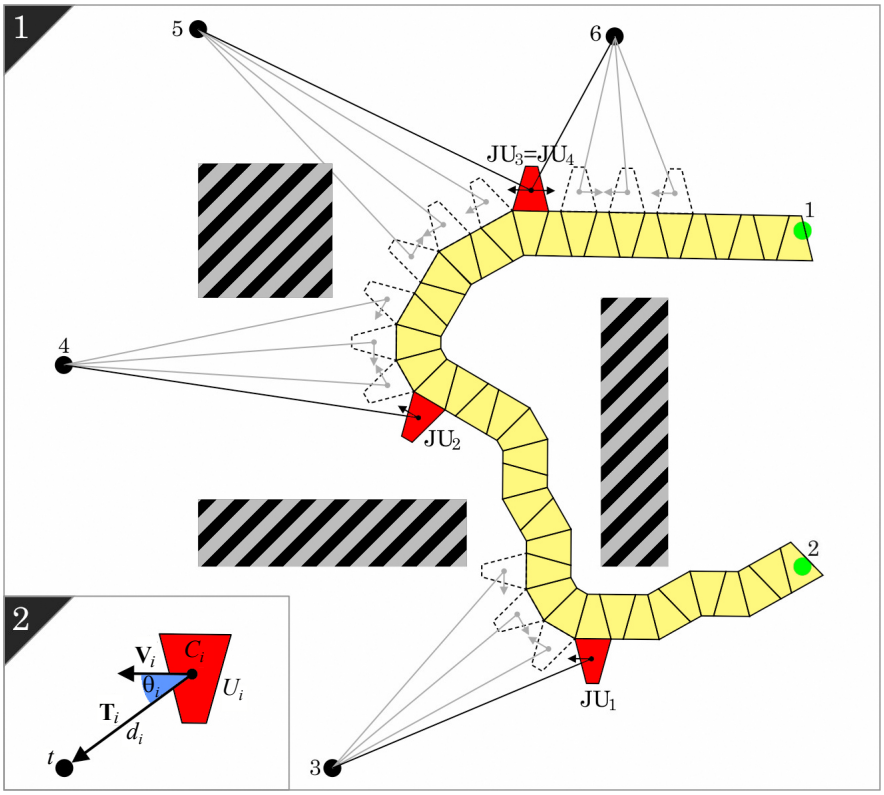

FIG. 8. Plan of a six-terminal TZ network.

$\mathbf{T}_i$ is the direction from a centroid (C_i) of the potential junction unit to a terminal t, a and b are parameters as in formula (1).

5) The junction units act as virtual starting terminals. The guide paths avoiding the obstacles are generated as shown in Fig. 9.1. The runs of the trusses are generated by aligning units to the guide paths according to formula (2) as shown in Fig. 9.2. The completed TZ network is shown in Fig. 9.3.

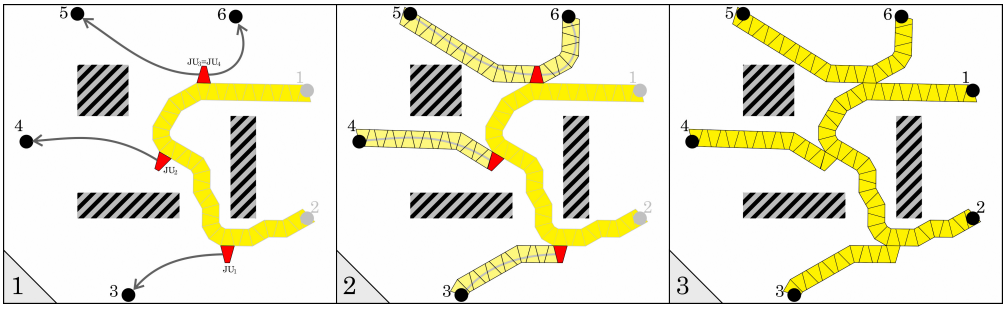

FIG. 9. The completion of a six-terminal TZ network.

2.2. Backtracking

Backtracking is a general algorithm suitable for finding solutions to some constraint satisfaction discrete problems [4]. It incrementally builds candidates to the solutions and abandons a candidate as soon as it determines that it cannot possibly be completed to an allowable solution [5, 6]. Here the procedure for creating TZPs can be formulated as follows:

1) At each step, a unit whose centroid is closer to the target (next terminal) is chosen.
2) If any point of the unit lies outside the allowed area or collides with another part of the TZP, the procedure steps back, switches the last module, and continues as in 1.
3) If any point of the truss module still lies outside the allowed area or collides with another part of the TZP, the procedure steps further back, switches the second last unit and continues as in 2 until all the constraints are satisfied.

The collisions are forbidden and are excluded by the "death penalty". The procedure produces allowable but rarely ideal solutions. The initial unit is fixed. Although the path tries to turn towards the Goal, there are cases where it continues along the obstacle in the "wrong" direction. Figure 10 shows some examples.

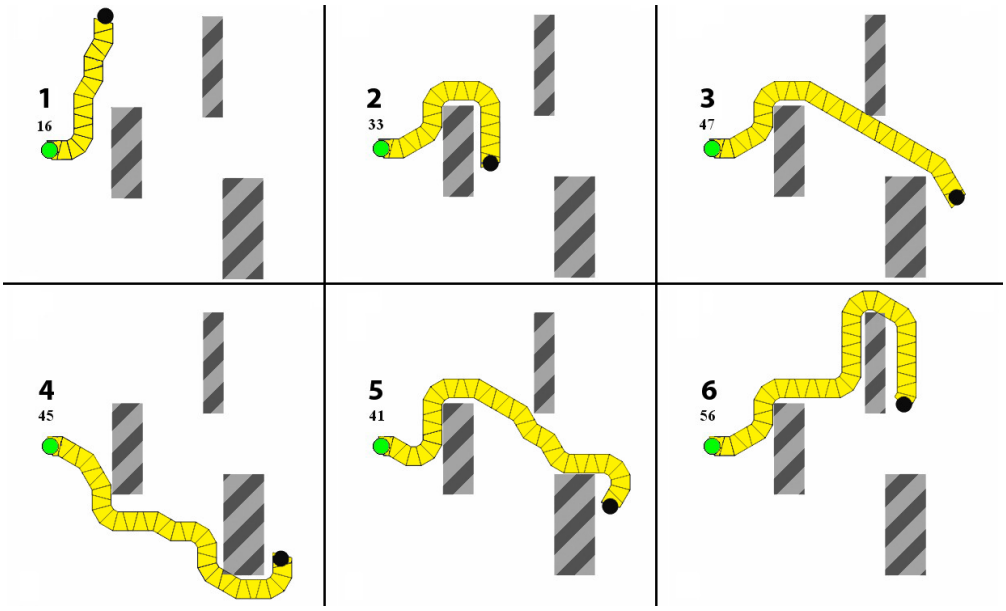

Fig. 10. Six cases of backtracking for a single run from Start to Goal terminals (indicated by green and black dots, respectively). The number of internal loops of the algorithm is indicated for each case.

This procedure has been expanded in order to generate multi-branch TZPs (see Fig. 11 for illustration):

1) The first TZP connecting the first two terminals is created as in a single-path case (Fig. 11.1).
2) If there is one terminal left – create a single path and stop.
3) The next two consecutive terminals are selected (here: 3 and 4).
4) An "available module", that is, a module from which a JU can be constructed, is selected where the sum of distances to the terminals 3 and 4 is minimal (Fig. 11.2).
5) A junction unit is created from the selected unit, and two independent paths to terminals 3 and 4 are constructed (Fig. 11.3).
6) Go to 2 (Figs 11.4 and 11.5).

Figure 11.6 shows the completed network of TZPs.

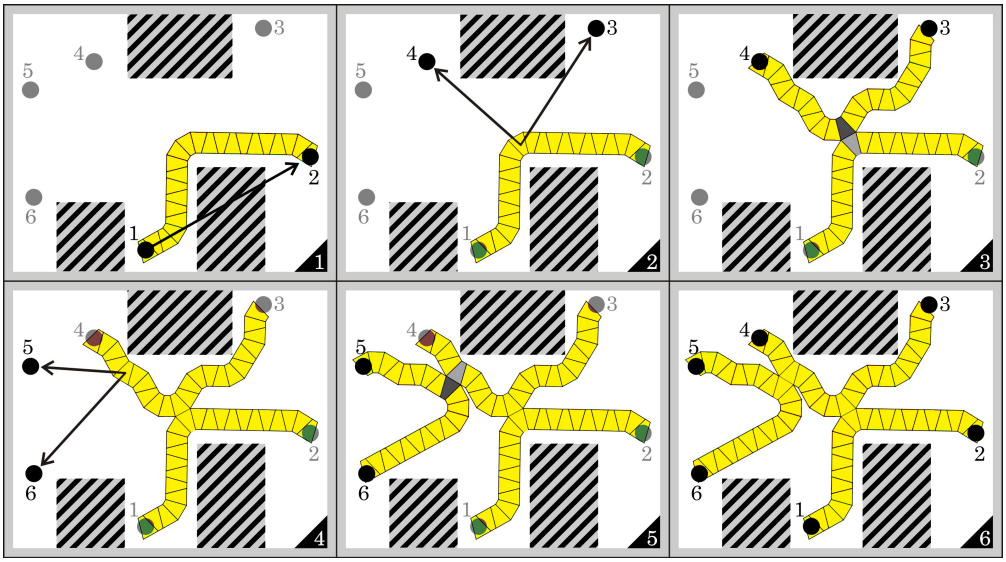

FIG. 11. An illustration of the backtracking procedure for creating a TZP network that links six terminals in a constrained environment.

For further details, including a formal description of the TZ module, its static analysis including topological properties using graph-theoretic methods, the degree of static indeterminacy of the TZ module and the proof of its rigidity using Cauchy's theorem see [7].

2.3. Evolutionary algorithms for a single-branch TZP

In order to apply the meta-heuristic optimization method, an objective function has been formulated [8]. Here it is a multi-objective cost function (CF_S)

since the problem has been defined as a minimization of: the number of TZ modules, the "reaching" error and collisions with the environment:

$$CF_S = G_S \times P_S, \tag{3}$$

where G_S evaluates how efficiently the TZP approaches the End Terminal (see Fig. 12.1) and P_S penalizes for violating the obstacles (see Fig. 12.2):

$$G_S = \frac{w_1 \sum_{i=1}^{m} d_i}{m} + w_2 \min\{d_1 ... d_m\} + w_3 m, \tag{4}$$

$$P_S = 1 + \frac{w_4 \sum_{k=1}^{U} c_k}{\sum_{k=1}^{U} A_k} \sum_{k=1}^{U} \frac{A_k}{1 + \mathrm{Log}_{3/2}[1 + \min\{d_1^k ... d_i^k ... d_m^k\}]}. \tag{5}$$

Figure 12 illustrates how G_S and P_S are calculated. Figure 12.1 corresponds to formula (4) and shows how distances d_i are calculated. Figure 12.2 corresponds to formula (5) and shows how distances d_i^k of the units that violate the k-th obstacle are calculated.

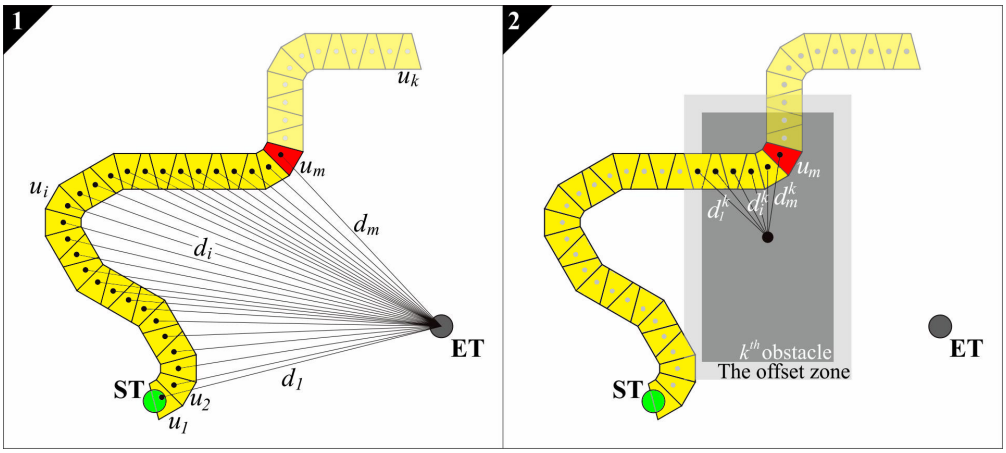

FIG. 12. The illustration of how G_S and P_S are calculated. The units' centroids are indicated by dots. ST and ET stand for: Start and End Terminals, respectively.

The operations of mutation and crossover have been developed and two classic meta-heuristics have been implemented: evolution strategy [9] (strong mutation without crossover) and genetic algorithm [10] (crossover of the best individuals at weak mutation rate). A number of parameter settings have been examined in

order to find the best solution for the reference setup. The results are collected in Fig. 13.

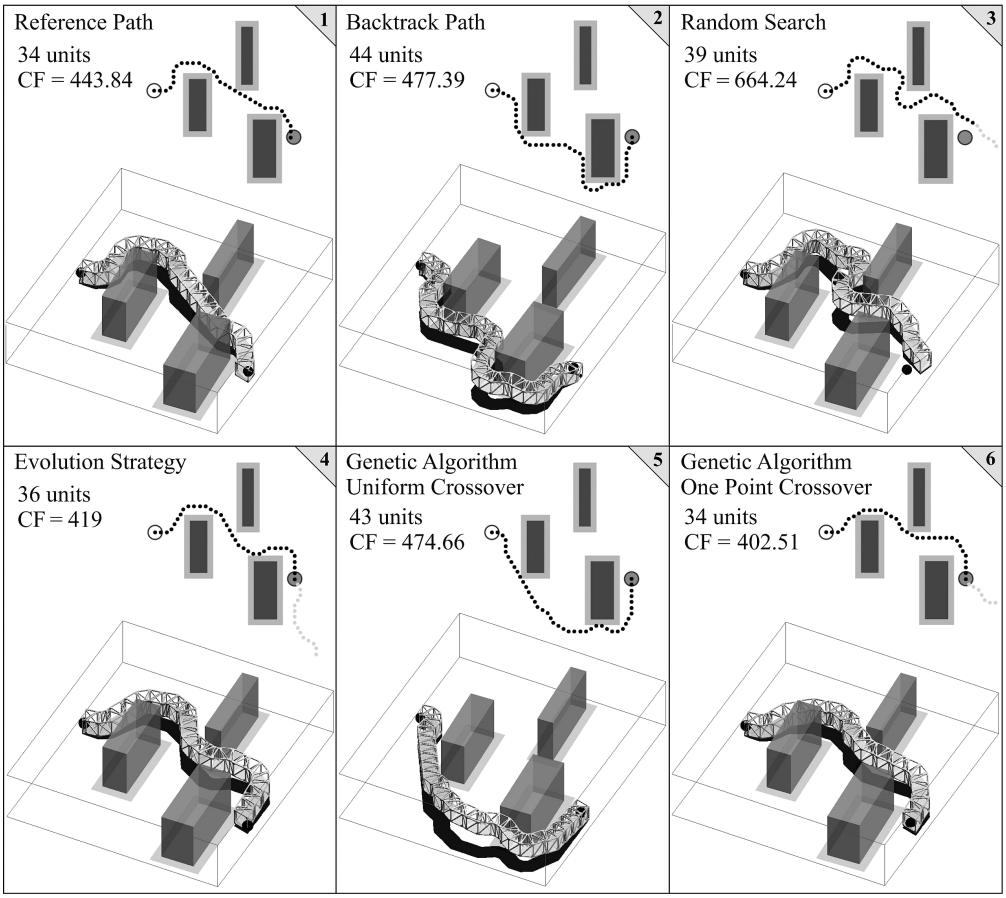

Fig. 13. TZPs linking two terminals in a constrained environment and constructed with various methods.

For further details on this method, including the nomenclature and coding of TZP into the genotype, its base-36 compression method and two additional validation experiments, see [11].

2.4. Graph-theoretic method for single-branch TZPs

The methods mentioned above produce allowable and relatively good results; however, they do not guarantee ideal solutions. The following method has been developed in order to find the ideal solutions for TZPs in given environments. It is based on TZP graphs. The search space of allowable, and in this case – optimal

path graphs can be represented by the "search space" binary tree (SST). This method resembles the Dijkstra's algorithm [12], but in fact it is fundamentally different, as there is no grid given beforehand. In the planar case, as in this example, a consecutive unit can be added in two possible ways. In such a case, the SST becomes a binary tree, as shown in Fig. 14.

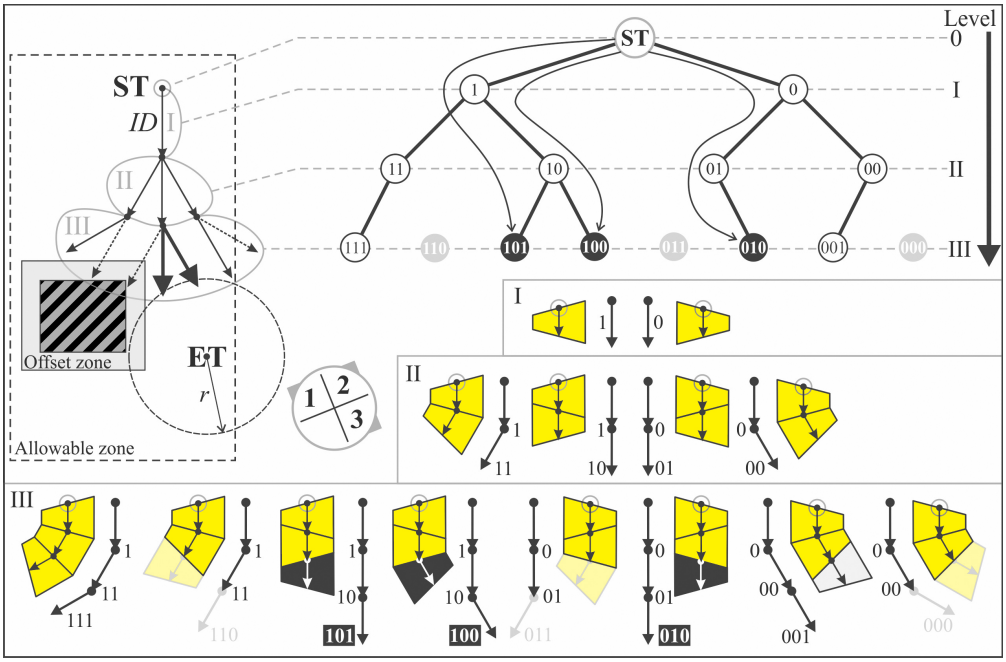

Fig. 14. An example of three alternative TZ path graphs: 1) Its framework in the real space; 2) the SST representing all allowable TZ path graphs. Three ideal paths are indicated; 3) All allowable TZPs corresponding to this SST.

The algorithm of creating the shortest path graphs from the ST to the ET based on breadth-first search and building consecutive levels of the SST is as follows:

1) Place units 1 and 0 according to the given initial direction (*ID*). For illustration see Fig. 14.
2) Prune the leaves that violate the constraints (obstacles and allowable zone AZ).
3) Check whether the leaves are within range r from ET.
   3.1) If yes, collect all the paths of SST, which start from the root (ST) and end in these leaves.
   3.2) If not, build the next level of SST. In other words, produce two branches from each leaf and go to 2.

This method strongly depends on the constraints, which must be intelligently applied. Otherwise the space of potential solutions becomes unmanageable [13]. Figure 15 shows 20 steps of the algorithm corresponding to 20 levels of SST.

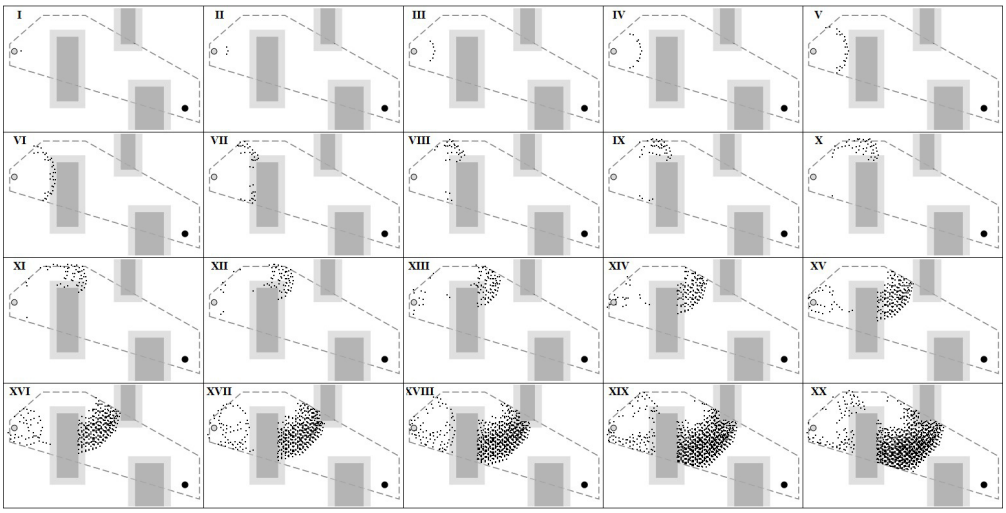

Fig. 15. The generation of all allowable nodes of SST. For clarity, only the nodes created at a particular step are shown. ST and ET are shown as gray and black discs, respectively. The allowable zone is shown as a dashed gray polygon.

The execution of the algorithm with these conditions and constraints took less than 18 minutes on an Intel Core7 PC and produced six different ideal TZPGs reaching ET with only 32 modules. This is fewer than the best result achieved previously. Figure 16 compares the results of the graph-theoretic approach for this experimental setup with the previous results (see Table 1): the reference "manual" solution (M) – 33 modules, genetic algorithm with uniform crossover (GA-UX) – 43 modules, genetic algorithm with one-point crossover (GA-OPX) – 34 modules, evolution strategy (ES) – 36 modules, backtracking (BT) – 44 modules, and random search (RS) – 39 modules.

For further details on this method, including other optimization criteria such as "geometric simplicity" and the "number of turns", and a case-study of retrofitting of an existing overpass with Truss-Z ramp composed of two sections of 144 modules in total (the visualization is shown in Fig. 1) see [14].

2.5. Effective optimization of a single-branch TZP with GPU

In this approach, image processing methods have been employed for qualitative evaluation of both types of collisions: TZP with the elements of environment and self-intersections of TZP. This method allows to selectively assign different

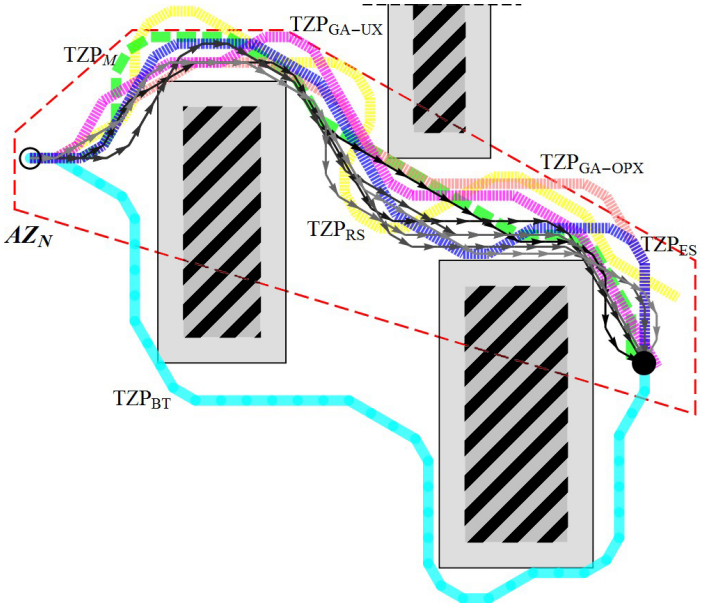

FIG. 16. Illustration of solutions produced by the graph-theoretic and previous approaches.

weights to different collisions with, e.g., trees, bushes, river, buildings, ground, etc. There are no restrictions on the fitness function definition. It can depend on any variable that can be represented by a two-dimensional map of any property of the environment. Figure 17 illustrates the efficiency of the algorithm in a simple, but non-trivial example: create a TZP from point A to B at minimal F in an environment with two obstacles. The result has been found within 15 evolution steps.

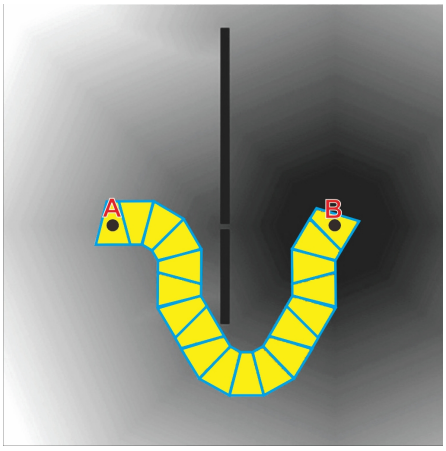

FIG. 17. A TZP from point A to B at minimal F. The black rectangles denote obstacles with infinite cost.

The algorithm has been parallelized [15] in *Mathematica*$^{\text{TM}}$ platform on four physical CPU cores and the *Wolfram Lightweight Grid*$^{\text{TM}}$ platform. Next, it has been implemented in the CUDA environment in order to compute the objective function taking advantage of highly efficient GPU processors. Figure 18 illustrates the range of 10^2–10^3 acceleration in comparison with the uncompiled *Mathematica*$^{\text{TM}}$ code.

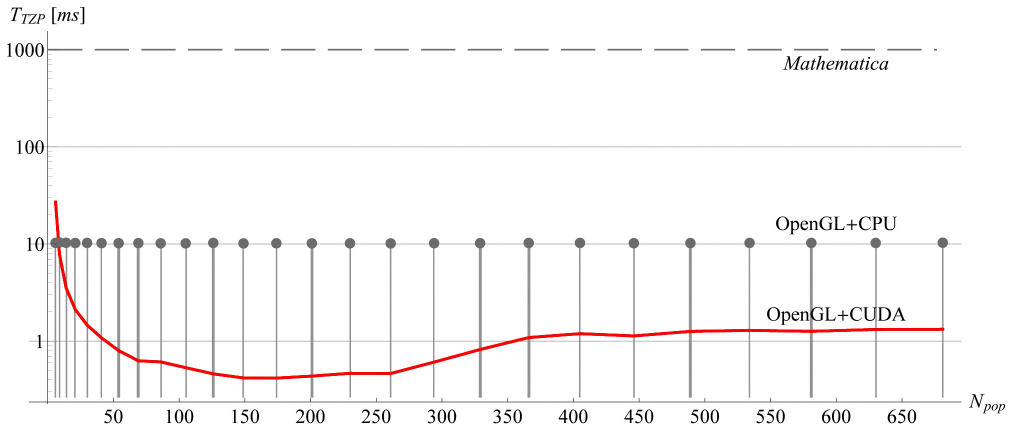

FIG. 18. The log-plot showing the dependence of the calculation time of the fitness function for a single genotype on the population size N_{pop}. The N_{pop} number of CUDA threads and serial CPU process are shown by the red line and gray dots, respectively.

The algorithm has been expanded with additional criteria, such as the costs of: earthworks, tree and bush removal, river crossing. Additionally, forbidden zones for placing the supports have been introduced, as well as the "geometric simplicity" criterion. Several case studies have been analyzed. Figure 19 illustrates four alternative solutions for creating a TZP on an existing slope. The height difference between ST and ET is over 10 m. Here, ET is defined not as a specific point but as a "destination" area along the top edge of the slope. A number of existing trees is to be preserved. Figure 19 shows four alternative solutions: 19.1 – minimization of the earthworks (at the expense of nine trees), 19.2 – preservation of all existing trees; 19.3 – the balance between earthworks and tree removal (only two trees are "shaved" but probably preservable); 19.4 – the earthworks and tree removal are minimal (removal of just two trees) and the straightness is maximized.

Figure 20 shows a physical model of the third TPZ from Fig. 19, that is, the solution that can be considered optimal as the earthworks and tree removal are minimal.

For further details on this method including other case-studies: "modular path in the garden" and "mountain pier" see [16].

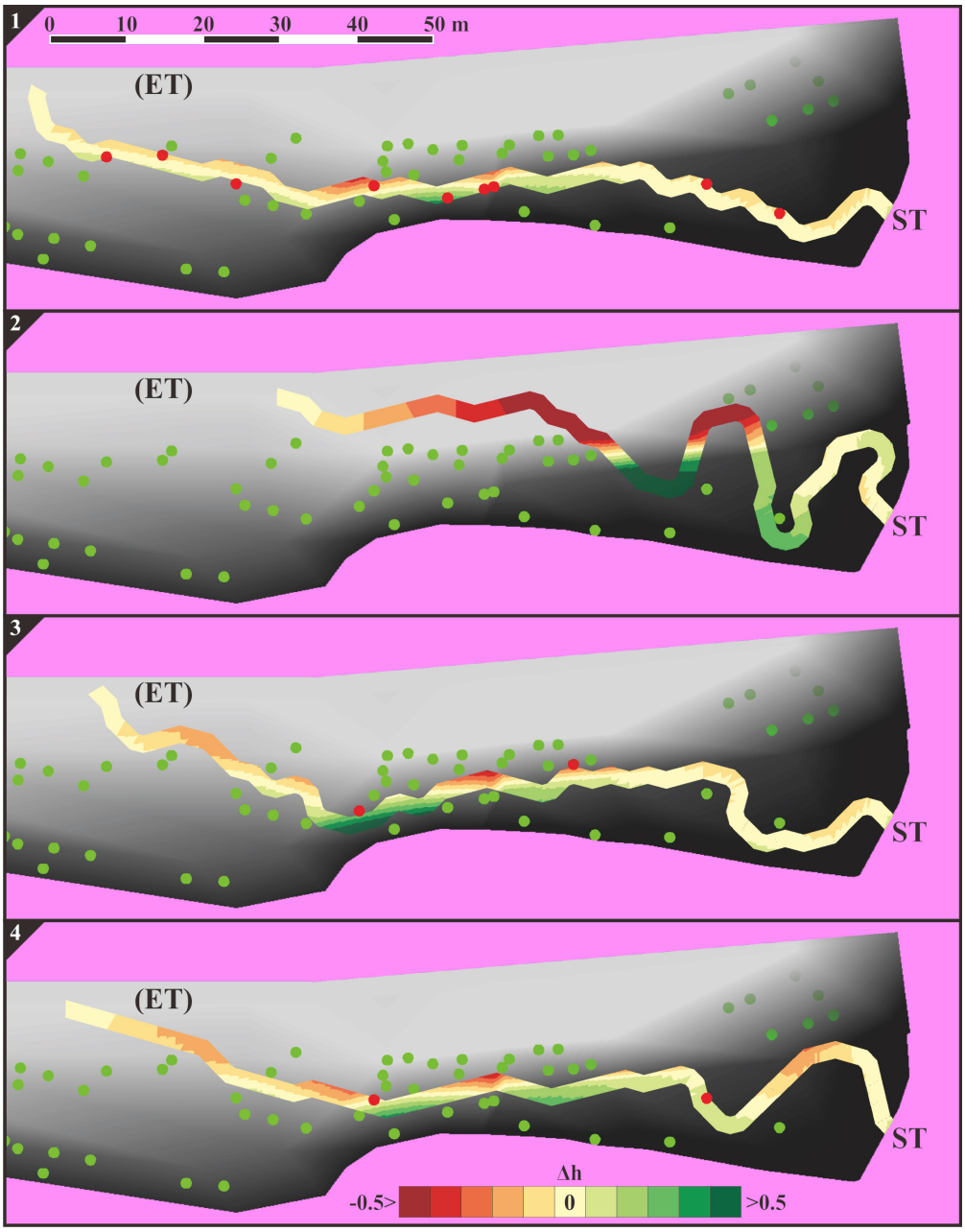

Fig. 19. Four alternative TZPs. The legend on the bottom reflects the "amount of earthworks": colors on the extreme left and right side correspond to removal and built-up of over 0.5 m of earth, respectively, at 0.125 m increments. The color in the middle of the legend means that TZ follows exactly the slope. Dots indicate the existing trees; red and green indicate trees to be removed and preserved, respectively.

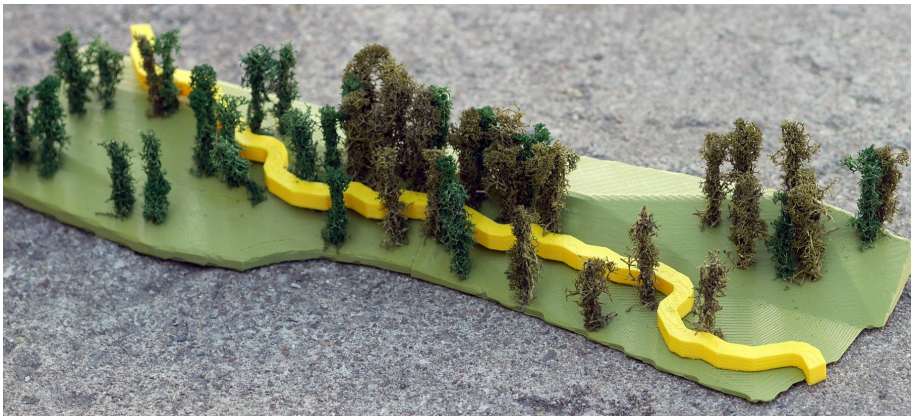

Fig. 20. A photograph of a physical scale model of the optimal TZP.

2.6. Evolution strategy for creating a multi-branch TZP

Previously mentioned heuristic methods were applied to single-branch TZPs. A new description of multi-branch Truss-Z paths (MTZ) has been developed (see Fig. 21).

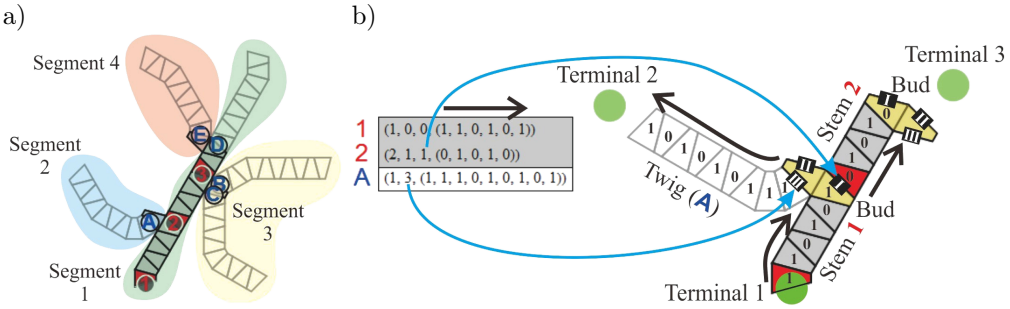

Fig. 21. a) An example of a six-terminal MTZ; b) a detail showing on the left partial genotype tabulated for clarity; on the right: the partial phenotype. Black arrows illustrate the direction of the sequence of trapezoidal units. The initial units of each stem, the buds and the twig are shown in red, yellow and white, respectively. The first unit of the first stem is rooted to Terminal 1. The blue arrows indicate corresponding bud faces.

For the optimization of MTZ, new modification operators have been introduced: random MTZ generator, degenerated MTZ fix, linear to matrix conversion, and a quantitative tabulated genotype comparison. Furthermore, four types of mutations have been introduced and it has been demonstrated that they must be applied jointly in order for the algorithm to converge properly. Figure 22 shows the result of evolution strategy-based algorithm applied for the optimization of an MTZ connecting six given terminals in space.

For further details on this method see [17].

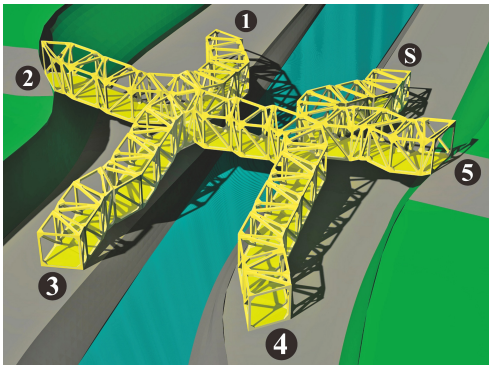

FIG. 22. A visualization of the MTZ produced by the evolution strategy-based algorithm. **S** stands for the ST.

2.7. Structural optimization of the TZ module

The methods mentioned above focused on the optimization of TZP, that is, the sequence of modules or a set of such sequences. In those optimizations, the modules were considered as strictly geometrical objects. Here a preliminary structural optimization of a TZ module is presented. The research on structural optimization of modular structures is relatively sparse and limited, and rare examples include optimization of bridges comprised of stacked rectangular truss panels [18], families of related products that share selected modules [19], and cell topology optimization in periodic structures [20, 21]. The method presented in [22] and summarized here is unique as it includes simultaneous optimization of module topology (configuration of the diagonal beams) and diameters of the structural elements. As the TZ system is intended to be universal, the subject of optimization is not a single global TZ structure, but rather all essentially unique global configurations. Here the assumed number of modules is $n = 2, ..., 7$. Such a formulation corresponds to the optimization of the worst-case configuration and allows the computations to be naturally parallelized. The 16 "worst-case" 7-module TZ configurations are shown in Fig. 23.

The objective function is the mass $m(\mathbf{x}, d)$ of a single module that directly depends on the vector $\mathbf{x}$ of beam diameters and the topological configuration d of the diagonals. The constraints, besides the natural lower bounds on the diameters, take the form of the upper bound on the von Mises (effective) stress $\boldsymbol{\sigma}^{\mathrm{eq}}$ in structural elements:

$$\boldsymbol{\sigma}^{\mathrm{eq}}(\mathbf{x}, d, S_n) := \max_{S \in S_n} \boldsymbol{\sigma}^{\mathrm{eq}}(\mathbf{x}, d, s) \leq 100 \text{ MPa}, \tag{6}$$

where s is a single global TZ configuration, while S_n is the set of all considered configurations of n elements. The optimization variables are denoted by d

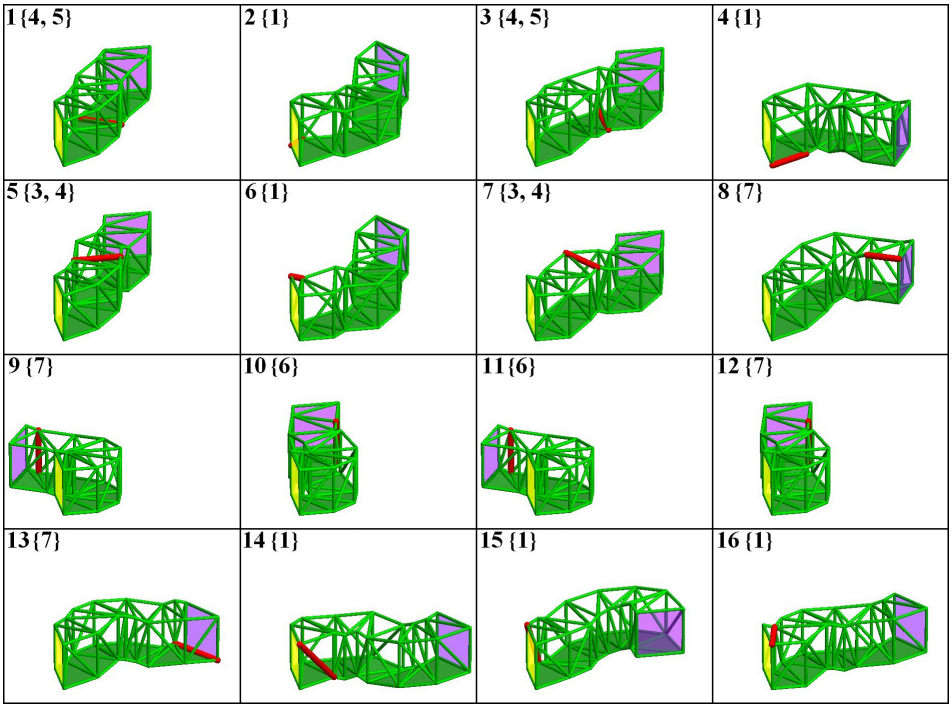

FIG. 23. The 16 "worst-case" 7-module TZ configurations. In each case, the indices of the most stressed beam and the corresponding module are shown. The most stressed beam is marked in red.

(topological configuration of the diagonal beams) and $\mathbf{x}$ (the vector of beam diameters). A typical distributed static design load of 5000 N/m^2 is assumed for the purpose of optimization [23].

The bound of 100 MPa is arbitrarily assumed as a stress value safe for steel. Because the optimization variables are of two kinds, a two-level optimization process is performed: the higher level optimizes the diagonal configuration d. For each such considered configuration, an independent lower level optimization is performed with respect to the beam diameters $\mathbf{x}$, see Table 1

$$\mathbf{x}(d, S_n) := \arg\min_{\mathbf{x}} m(\mathbf{x}, d), \tag{7}$$

where $\mathbf{x}(d, S_n)$ is the vector of the optimal beam diameters for the topology d and configuration set S_n.

Topology optimization has a discrete character, and because of the relatively small number of possible configurations, it is carried out using an exhaustive search. The lower level optimization (with respect to the beam diameters $\mathbf{x}$) is performed using a local evolution approach with an adaptive definition of the neighborhood: the neighborhood is dynamically modified during the opti-

Table 1. Scheme of the optimization procedure.

For each considered set $S_n \in S$:

1. Select the initial vector $\mathbf{x}$ of the beam diameters.
2. For each configuration d of the diagonals:
   (a) Compute the mass $m(\mathbf{x}, d)$ of the module.
   (b) Compute the local stiffness matrix and the local load vector of a single module.
   (c) For each configuration $s \in S_n$
      i. Aggregate the local stiffness matrix and the local load vector of single modules into the global stiffness matrix and the global load vector of the configuration s.
      ii. Compute the global displacement vector and the compliance $C(\mathbf{x}, d, s)$.
      iii. For each beam, compute the local nodal reaction forces and use the circular hollow section formulas to compute the distribution of the von Mises stress and its maximum value. Store the maximum values, as found for each beam of the module, in the vector $\boldsymbol{\sigma}^{\mathrm{eq}}(\mathbf{x}, d, s)$.
   (d) Take the maximum of $C(\mathbf{x}, d, s)$ for all $s \in S_n$ to find the maximum (worst-case) compliance $C(\mathbf{x}, d, S_n)$.
   (e) Take the maximum of the successive elements of $\boldsymbol{\sigma}^{\mathrm{eq}}(\mathbf{x}, d, s)$ for all $s \in S_n$ to find the vector $\boldsymbol{\sigma}^{\mathrm{eq}}(\mathbf{x}, d, S_n)$ of the maximum von Mises stresses of the beams.
   (f) Check the stress constraints. If not satisfied, update $\mathbf{x}$ and go to Point 2a.
   (g) Check the stop conditions. If not satisfied, update $\mathbf{x}$ and go to Point 2a.
   (h) Store the optimum $\mathbf{x}(d, S_n)$, the corresponding mass of the module $m(d, S_n)$, the worst-case compliance $C(d, S_n)$ and the maximum von Mises stresses of the beams $\boldsymbol{\sigma}^{eq}(\mathbf{x}, d, S_n)$.

mization process in order to intensify the search in the vicinity of the active constraints.

For more details on this method see [22]. A further extension, including the geometry of the module, is presented in [24]. The presented formulation is general enough to consider in future research a number of related, open and timely problems, such as decentralized and distributed control of modular structures [25, 26] or local health monitoring of modular structures [27].

3. Conclusions

This post-workshop paper presented an overview of various computational optimization methods applied to the Truss-Z system. Some of the methods presented are relatively simple and straightforward to implement, such as alignment of the Truss-Z modules to given guide path or backtracking. They produce results relatively quickly and are applicable to the problems of a rather smaller size. They may produce satisfactory results, but this is not guaranteed. Other methods, such as graph-theoretic or evolutionary algorithms, require more elab-

oration but provide very good results. Taking advantage of well-established and very effective computational tools, such as image processing and parallelization combined with heuristic methods, produces very good results in optimizations of Truss-Z paths of real-life size and complexity.

The purpose of this paper was to demonstrate the potential of computational optimization methods and their usability in the field of architecture-related design by showing relatively concrete examples and problems.

Acknowledgements

This work was completed as part of the research project no. 2019/33/B/ST8/02791 funded by the National Science Centre, Poland.

References

1. R.E. Smith, *Prefab architecture: a guide to modular design and construction*, John Wiley & Sons, Hoboken, NJ, 2011.
2. M. Zawidzki, *Discrete optimization in architecture – Extremely modular systems*, Springer Briefs in Architectural Design and Technology, Springer Singapore, 2017, doi: 10.1007/978-981-10-1109-2.
3. M. Zawidzki, Selected applications of computational intelligence methods to optimization in architecture, [in:] *Workshop on Engineering Optimization*, November 4th 2019, Institute of Fundamental Technological Research, Polish Academy of Sciences, Warsaw, Poland, 2019, http://bluebox.ippt.pan.pl/~ptauzow/WEO/presentations/Prezentacja%20MZ.pdf.
4. P. van Beek, Backtracking search algorithms, [in:] *Foundations of artificial intelligence*, F. Rossi, P. van Beek, T. Walsh [Eds], Vol. 2, pp. 85–134, Elsevier, Amsterdam, 2006.
5. D.E. Knuth, *The art of computer programming*, Addison-Wesley, Reading, Mass., 1968.
6. E. Gurari, CIS 680: DATA STRUCTURES: Chapter 19: Backtracking algorithms, 1999, online course available at https://web.archive.org/web/20070317015632/http://www.cse.ohio-state.edu/~gurari/course/cis680/cis680Ch19.html#QQ1-51-128.
7. 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, doi: 10.1016/j.advengsoft.2011.12.012.
8. Z. Michalewicz, D.B. Fogel, *How to solve it: modern heuristics*, Springer Verlag, Berlin, 2013, doi: 10.1007/978-3-662-07807-5.
9. I. Rechenberg, *Evolution strategy – optimization of technical systems according to the principles of biological evolution* [in German: *Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*], PhD thesis, 1971.
10. M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA, 1996.
11. 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, doi: 10.1016/j.advengsoft.2013.04.022.
12. E.W. Dijkstra, A note on two problems in connexion with graphs, *Numerische Mathematik*, **1**: 269–271, 1959, doi: 10.1007/BF01386390.

13. D.R. Morrison, S.H. Jacobson, J.J. Sauppe, E.C. Sewell, Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning, *Discrete Optimization*, **19**: 79–102, 2016, doi: 10.1016/j.disopt.2016.01.005.
14. M. Zawidzki, Retrofitting of pedestrian overpass by Truss-Z modular systems using graphtheory approach, *Advances in Engineering Software*, **81**: 41–49, 2015, doi: 10.1016/j.advengsoft.2014.11.004.
15. A. Gottlieb, G.S. Almasi, *Highly parallel computing*, Benjamin/Cummings Publishing, Redwood City, CA, 1989.
16. M. Zawidzki, J. Szklarski, Effective multi-objective discrete optimization of Truss-Z layouts using a GPU, *Applied Soft Computing*, **70**: 501–512, 2018, doi: 10.1016/j.asoc.2018.05.042.
17. M. Zawidzki, Optimization of multi-branch Truss-Z based on evolution strategy, *Advances in Engineering Software*, **100**: 113–125, 2016, doi: 10.1016/j.advengsoft.2016.07.015.
18. A. Tugilimana, A.P. Thrall, R.F. Coelho, Conceptual design of modular bridges including layout optimization and component reusability, *Journal of Bridge Engineering*, **22**(11), 04017094, 2017, doi: 10.1061/(ASCE)BE.1943-5592.0001138.
19. B.R. Torstenfelt, A. Klarbring, Structural optimization of modular product families with application to car space frame structures, *Structural and Multidisciplinary Optimization*, **32**(2): 133–140, 2006.
20. E. Moses, M.B. Fuchs, M.B. Ryvkin, Topological design of modular structures under arbitrary loading, *Structural and Multidisciplinary Optimization*, **24**(6): 407–417, 2002.
21. C.W. Zhou, J.P. Lainé, M. Ichchou, A.M. Zine, Multi-scale modelling for two-dimensional periodic structures using a combined mode/wave based approach, *Computers & Structures*, **154**: 145–162, 2015.
22. M. Zawidzki, Ł. Jankowski, Optimization of modular Truss-Z by minimum-mass design under equivalent stress constraint, *Smart Structures and Systems*, **21**(6): 715–725, 2018, doi: 10.12989/sss.2018.21.6.715.
23. *LRFD Guide Specification for the Design of Pedestrian Bridges*, American Association of State Highway and Transportation Officials (AASHTO), Washington, DC, 2009.
24. M. Zawidzki, Ł. Jankowski, Multiobjective optimization of modular structures: weight versus geometric versatility in a Truss-Z system, *Computer-Aided Civil and Infrastructure Engineering*, **34**(11): 1026–1040, 2019, doi: 10.1111/mice.12478.
25. D. Pisarski, R. Konowrocki, Ł. Jankowski, Scalable distributed optimal control of vibrating modular structures, *Structural Control and Health Monitoring*, **27**(4), e2502-1-21, 2020, doi: 10.1002/stc.2502.
26. B. Popławski, G. Mikułowski, A. Mróz, Ł. Jankowski, Decentralized semi-active damping of free structural vibrations by means of structural nodes with an on/off ability to transmit moments, *Mechanical Systems and Signal Processing*, **100**: 926–939, 2018, doi: 10.1016/j.ymssp.2017.08.012.
27. J. Hou, Ł. Jankowski, J. Ou, Frequency-domain substructure isolation for local damage identification, *Advances in Structural Engineering*, **18**(1): 137–153, 2015, doi: 10.1260/1369-4332.18.1.137.

Received April 24, 2020; revised version August 21, 2020.