

An Active Structure that Learns

Bernd Domer¹ and Ian F. C. Smith, F.ASCE²

Abstract: Tensegrity structures are composed of cables and struts that become stable through self stress. They are good candidates for implementation of active structural control because their flexibility may mean that they cannot meet serviceability criteria. Changes to the self stress influence the form of the structure. A reliable closed-form solution for obtaining control commands for telescopic compression elements in order to obtain a required shape does not exist for such a closely coupled and geometrically nonlinear structure. Simulating the structural behavior after all possible control commands and testing against constraints and the objective function requires computational times that grow exponentially with the number of actuators. This paper demonstrates that search time can be reduced through use of stochastic search methods and that incrementally storing successfully applied control commands in a case-based reasoning system increases performance during service lives (learning). Such results demonstrate that enhancing control with advanced computing methods provides opportunities for innovative structures.

DOI: 10.1061/(ASCE)0887-3801(2005)19:1(16)

CE Database subject headings: Active control; Structural control; Stochastic process; Struts; Cables.

Introduction

Although active structures may increase load-carrying range, they have mainly been conceived to protect structures against earthquakes and to enhance occupant comfort under high wind loading. Ensuring control reliability over long return periods is difficult and expensive. Combining innovative structures, such as tensegrities, with active structural control of serviceability criteria involves new challenges and interesting opportunities. Tensegrities are a subclass of cable structures, where compression members are held apart by a network of tension members. The main distinction from traditional cable structures is that tensile forces are not necessarily anchored. Instead, they are equilibrated by self-stress states. Variations of self stress are used to control the shape of a tensegrity structure under loading (Fest et al. 2003). Although deflections of a tensegrity with a given geometry and loading can be calculated, doing the inverse to obtain control commands for telescopic bars to satisfy a form objective is much more difficult.

While tensegrities have been an area of interest for researchers for some time (Motro 1992; Williamson and Skelton 1998), their precise definition is still controversial. The most recent definition has been given by Motro and Raducanu (2001):

“A tensegrity is a system in a stable, self-equilibrated state that contains a discontinuous set of components in compression inside a network of components in tension.”

¹Dr ès sc., Project Manager, TEKHNE Management SA, Ave. de la Gare 33, 1003 Lausanne, Switzerland.

²Professor, Structural Engineering Institute, IMAC-ENAC, EPFL, 1015 Lausanne EPFL, Switzerland. E-mail: ian.smith@epfl.ch

Note. Discussion open until June 1, 2005. Separate discussions must be submitted for individual papers. To extend the closing date by one month, a written request must be filed with the ASCE Managing Editor. The manuscript for this paper was submitted for review and possible publication on November 13, 2003; approved on March 15, 2004. This paper is part of the *Journal of Computing in Civil Engineering*, Vol. 19, No. 1, January 1, 2005. ©ASCE, ISSN 0887-3801/2005/1-16-24/\$25.00.

Designing tensegrities is not a routine task in structural engineering. Design codes and guidelines are incomplete. Two additional challenges are as follows:

1. **Form finding.** The initial equilibrium position of the self-stressed structure has to be determined by either experimental or analytical methods before analysis. This process is called form finding.
2. **Geometrical nonlinearity.** The assumption of small deflections is not usually valid. Therefore, equilibrium conditions have to be formulated using the geometry of the deformed structure.

Dynamic relaxation is the most attractive analysis method for tensile and tensegrity structures (Barnes 1977). It is capable of modeling nonlinear geometric and material behavior. This iterative method traces the motion of structural nodes until the nodes converge to an equilibrium state. Therefore form finding and analysis are carried out together. Furthermore for cable structures, calculations are performed rapidly when compared to more usual methods since no matrix inversion is required.

Recently, Murakami (2001a,b) presented equations for static and dynamic analysis of tensegrities. A nonlinear dynamic analysis methodology is described in Kahla et al. (2000). Sultan et al. (2002) provided formulations and an analytical basis for dynamic analysis of tensegrity structures. Node friction is identified as the source of inaccuracies during simulation. Although analytical results are presented, they have not been validated on a full-scale structure.

Averseng et al. (2002) demonstrate a methodology to calibrate a tensegrity grid such that a targeted self-stress state is attained. Results are verified on a physical model. The impact of geometrical nonlinear behavior is neglected. This paper is one of the rare cases where simulated behavior is compared with the real behavior of tensegrity structures. Oppenheim and Williams (2001) relate the nonlinear stiffening of a three-bar tensegrity structure to applied torque. They state that vibrations of such a system cannot be suppressed by prestress or by active control. A further observation is that additional friction effects exist mainly at the joints in real systems.

Variations of self stress of tensegrities change their form. This property can be used for structural control purposes in order to adapt the structure to changing environments. Djouadi et al. (1998) discuss optimal control strategies to tensegrity structures. Weight matrices used in this approach are difficult to determine. During simulations, damping of structural vibrations is the control objective. The system is controlled by virtually changing the bar and cable lengths. No experimental verification of this approach has been reported. Also constraints that limit cable stresses are not introduced.

Sultan (1999) also proposes a formulation for tensegrity structure control. The control application is illustrated using the example of an aircraft motion simulator. The goal is to minimize error between deployment path and equilibrium path. This is a conventional control approach for deployable structures. To derive equations, multiple mechanical effects are neglected. No verification on a full-scale structure has demonstrated that the assumed linearizations are appropriate.

Skelton et al. (2000) conclude that since only small amounts of energy are needed to change the shape of tensegrity structures, they are advantageous for active control. Proposed applications are airplane wings and for microsurgery. Active components in these proposals are cables.

This paper describes a study of a unique tensegrity structure that contains telescopic compression members and is actively controlled using advanced computing methods. Behavior and control aspects are reviewed in the following section. "Case-Based Reasoning for Control" presents the application of case-base reasoning to improve performance of the control system and "Evaluation of Case-Base Reasoning" contains test results related to various aspects of case-based reasoning. The most important demonstration in this section is that the structure learns through incremental addition of cases obtained during previous loading events.

Behavior and Control of Tensegrity Structure

Tensegrities have been analyzed in many ways. While all proposed methods have appropriate uses, applications to full-scale structures reveal additional challenges. Some assumptions (mainly linearizations) have not been validated experimentally. Also model accuracy may not be sufficient for applications related to active structures. This section summarizes previous work

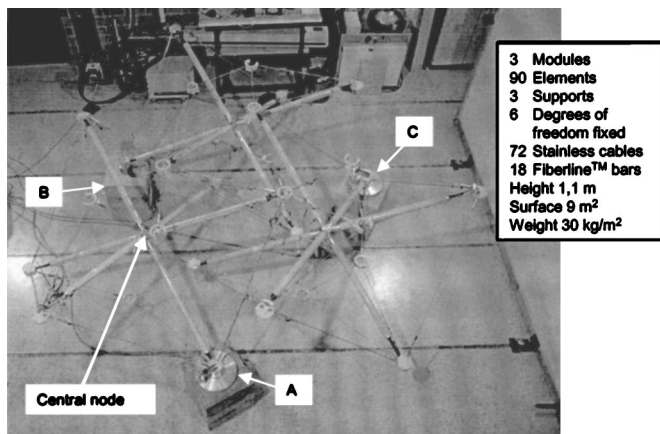


Fig. 1. Initial three-module structure

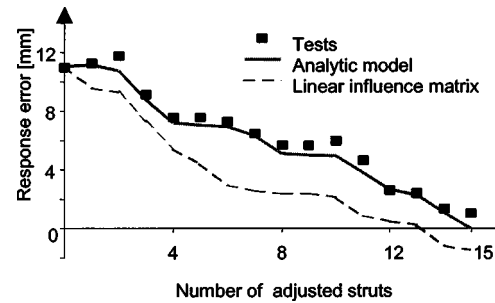


Fig. 2. Nonlinear behavior during strut adjustments, from Fest et al. (2003)

at EPFL, Lausanne, Switzerland. This work has involved the construction of a full-scale tensegrity structure (Fig. 1) and an initial validation of the control methodology (Fest, personal communication 1999).

The computational model has been compared with measurements (Y. Perelli, personal communication 2000). Nonlinear behavior precludes linear superposition even for small control movements, see Fig. 2. The use of an influence matrix and superposition leads to inaccurate predictions of the ability of strut adjustments adjustment to correct for slope deviations, noted as "response error" in Fig. 2.

Modeling inaccuracies due to joint friction may lead to divergent behavior during active structural control. A hybrid method that increases accuracy of dynamic relaxation results by a subsequent neural network run has successfully been tested with two different structural configurations (Domer et al. 2003). The neural network was trained to compensate for effects that are not included in the computational model. Fig. 3 shows the results of the most successful network configuration. Values above unity on the vertical axis represent accuracy increases.

Inspired by an initial design of the design office Passera and Pedretti, Fest (2002) designed and constructed a modular full-scale tensegrity structure equipped with telescopic bar devices for active structural control. Each module consists of 24 cables and six bars. Bars meet at the center of a module at the central node. In the latest version, a five-module structure has been constructed (see Fig. 4).

Two bars of each module have been motorized to be coupled with the control system. The control system thus acts on a total of

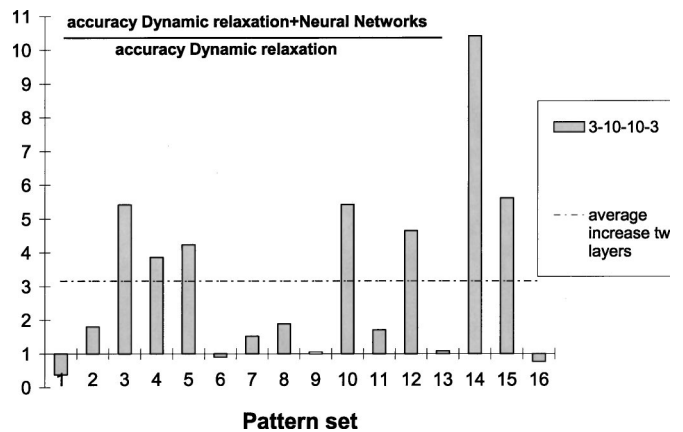


Fig. 3. Using neural network to increase accuracy of computational model

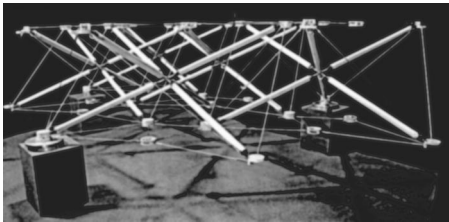


Fig. 4. Five-module tensegrity structure without actuators on struts

ten motorized telescopic bars in the five module structure (see Fig. 5).

A serviceability criterion requiring a constant slope from the region of Node 43 to the line marked by Nodes 37 and 48 is the control objective. Fig. 6 shows a plan view of the structure. The chain lines indicate a triangle with these nodes at vertices. Telescopic bars are shown using thick continuous lines. Circled node numbers are loaded nodes and boxed node numbers are nodes used for the distance metric that is described in a subsequent section [Eq. (2)].

The control objective means that the initial slope parameter under dead load of the system, named $slope_{initial}$ in Fig. 7, is maintained constant. The constant, 100, in this slope parameter is a magnification factor. The real slope would require division by the distance between Node 43 and the line joining Nodes 37 and 48. Since this value is a constant for all tests, it was omitted from the calculations. The cost-function [Eq. (1)] uses this value

$$cost = slope_{initial} - 100 \cdot \left| h_{43} - \frac{h_{48} + h_{37}}{2} \right| \quad (1)$$

The values, h_{43} , h_{48} , and h_{37} , =vertical positions of Nodes 43, 48, and 37, respectively. There is no closed-form solution for determining bar movements using required slope as input. Also iterative gradient search methods are not reliable since many local minima are present and their occurrence is amplified by the closely coupled nonlinear behavior of tensegrity systems. As a result, commands are generated and this is followed by a structural analysis and finally a test of the requirements (generate, analyze, and test). Testing all possibilities is not possible due to the combinatorial nature of the task. For example, for 0.1 mm increments of movement over 50 mm, ten actuators have 500^{10} control possibilities. Even when structural analyses take 0.1 s (making each complete iteration take up to 0.3 s) all possibilities could require 9.3×10^{16} centuries to generate, analyze, and test. A

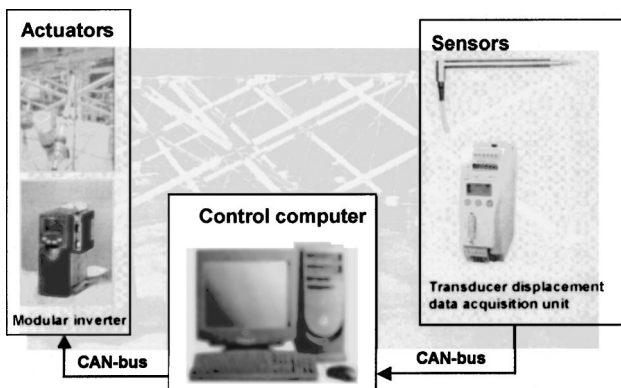


Fig. 5. Active control components

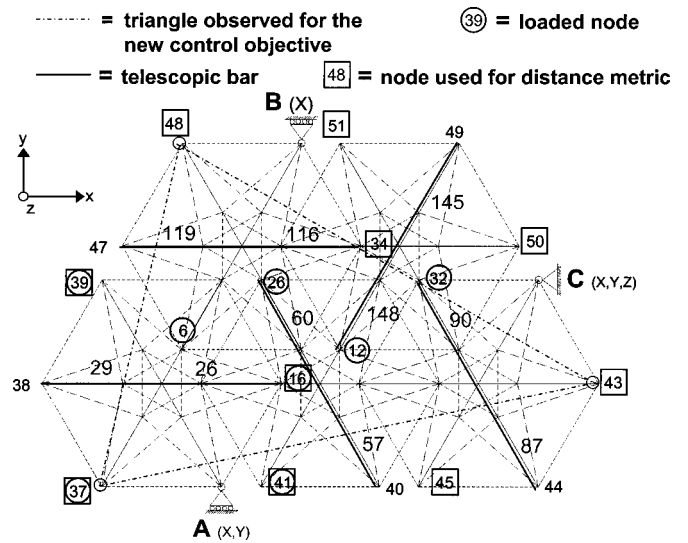


Fig. 6. Five-module structure with loaded nodes and controlled struts; A, B, and C are supports

stochastic search is useful for such situations. Simulated annealing was initially tested as a stochastic search method to find good control commands. Such commands compensated deflections caused by external loading (Fest et al. 2003). A definition of search is given in Leake (2001) as follows:

“Search is a process of formulating and examining alternatives. It starts with an initial state, a set of candidate actions, and criteria for identifying the goal state. [...] Starting from the initial state, the search process selects actions to transform that state into new states, which themselves are transformed into more new states, until a goal state is generated.”

Although one objective of search is to converge as fast as possible to the optimal value, another is to visit a sufficient number of candidate solutions to avoid local minima. Salama et al. (1993) proposed the use of stochastic search in conjunction with a structural control task. Although the method used (simulated annealing) identified a set of good control commands, the cost of the analytical solution differed significantly from the measured response of the actively controlled system. Control movements induced deflections that were in the magnitude of microns. These deflections were the result of structural nonlinear behavior. The linear model used to evaluate the objective function encountered inaccuracies during the search process.

Shea et al. (2002) proposed a system for intelligent structural control of tensegrity structures (see Fig. 8).

This paper provides details of the implementation as well as

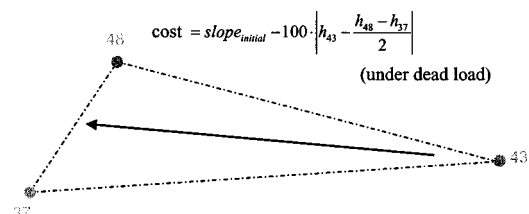


Fig. 7. Control objective

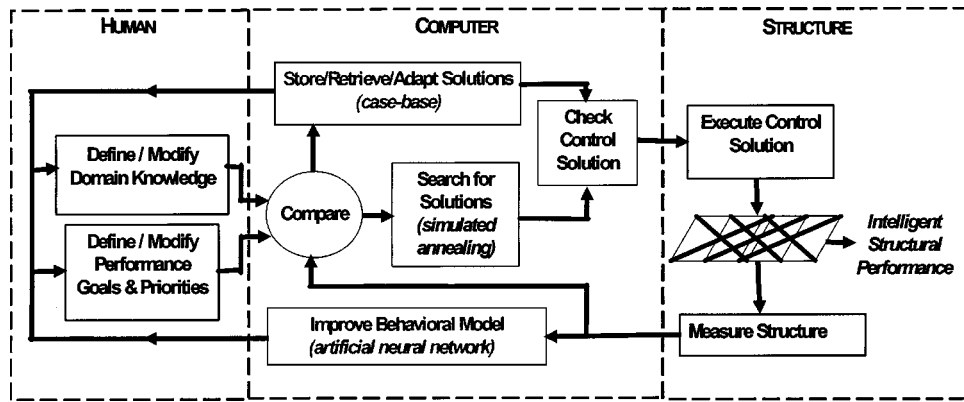


Fig. 8. Schema for intelligent structural control of tensegrity systems, from Shea et al. (2002)

experimental verification of a limited number of elements in this schema.

Comparison of different search techniques applied to the same task exist (El-Beltagy and Keane 1999; Manoharan and Shanmuganathan 1999; Connor and Shea 2000). Nevertheless, advantages are not task independent. Wolpert and Macready (1997) propose the “no free lunch theorem” for optimization algorithms that do not use problem-specific tuning. This theorem states that algorithms that perform well for one class of tasks do not necessarily produce good results for other classes. Generally, no one algorithm is best for all classes. Therefore, engineering studies are needed in a range of applications to determine the most suitable match between algorithm and task. Techniques for identifying control commands were studied by Domer (Domer 2003; Domer et al. 2003) who compared the following algorithms:

- Simulated annealing;
- Probabilistic global search Lausanne (PGSL); and
- Genetic algorithms (GAs).

Simulated annealing stems from the analogy of cooling metals. Temperature schedules are used to control the arrangement of atoms during their crystallization process. It is a stepwise technique that allows moves to inferior solutions and is therefore able to overcome local minima (Dowland 1995). Probabilistic global search lausanne is a newly developed technique and is based on the assumption that sets of better values are more likely to be found in the neighborhood of sets of good values and, therefore, intensifies search in regions that contain sets of good solutions. Search is driven by probability density functions. Gradients are not required (Raphael and Smith 2003). Genetic algorithms are inspired by Darwin’s theory of evolution. It observes that nature produced highly adapted creatures over a long process and only the fittest had the possibility to reproduce themselves (Goldberg 1989).

Fig. 9 shows typical convergence behavior. Several iterations may be necessary before good solutions are identified. The success of this search does not necessarily require solutions that are near to 0. It may not be possible to counteract completely all deflections within the constraints of this task. Furthermore, the usual inaccuracies between behavioral models and real behavior often do not justify the computational cost of a theoretically better solution that provides improvements below a practical threshold. Such tradeoffs help determine the most appropriate levels of accuracy.

Employing stochastic search to find good control solutions provides flexibility regarding control objectives and constraints since they can easily be changed. Nevertheless, calculation and

search time is still in the region of hours and this is too long even for applications in quasistatic control. Although calculation and search time decrease with more powerful computers, the exponential computational complexity of the control task precludes attaining acceptable times through increases in computer speed. This means, for example, that adding modules, telescopic bars, new, or multiple control objectives increases demand for computational power much faster than expected increases in processor speed.

Case-Based Reasoning for Control

Case-Based Reasoning Methodology

Case-based reasoning (CBR) systems build on the observation that previous experience is useful. Humans solve new situations by first searching their memory for similar tasks they have successfully solved in the past. Retrieved solutions are then adapted (Kolodner 1993, Leake 1996). The core component of CBR systems is the case base, where past experience is stored.

Cases are stored as pairs of task–solution descriptions. An interesting characteristic of CBR systems is that by storing successful cases in the case base, they improve performance over time. However, by storing more and more cases in the case base, the administrative overhead increases and system performance subse-

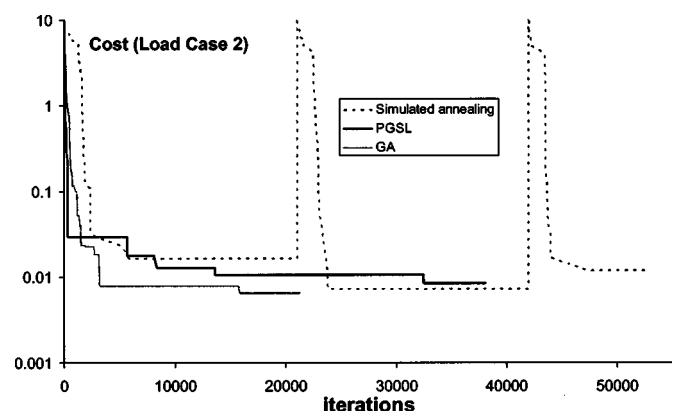


Fig. 9. Best-so-far curve load Case 2

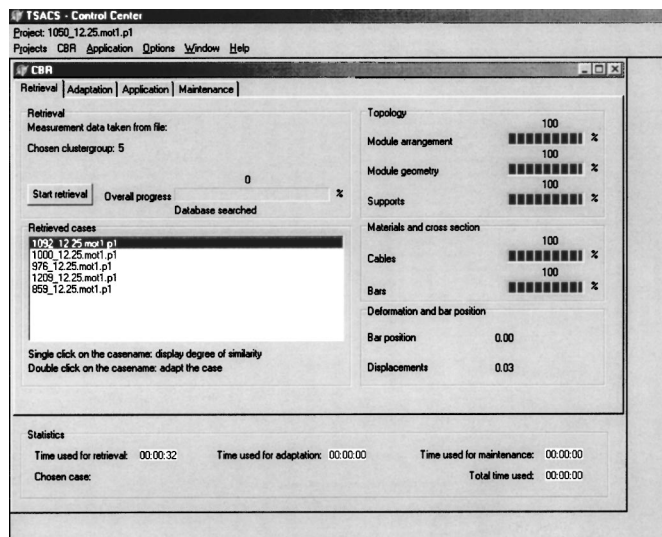


Fig. 10. Screen shot of retrieval module

quently decreases without increasing competence. Key tasks in the development of a CBR system include the following activities:

- Design of the case base;
- Choice of similarity measurement for case retrieval;
- Selection of the case adaptation methodology; and
- Development of case management strategies.

Case-base design involves the choice of representation techniques and case memory organization. Several representation schemes are available. If cases are used for human browsing alone, text and image representations are sufficient. However, if cases need to be adapted automatically, representations need to accommodate the requirements of the adaptation code. A simple representation involves a fixed set of attributes and values similar to that in a relational database. Object representations containing decompositions and abstraction hierarchies are also common.

Case memory organization affects efficiency and ease of retrieval. A flat list organization is sufficient for relatively small case bases. Hierarchical organizations improve the efficiency of retrieval when the size of a case base increases. Clustering also speeds up retrieval. A more in-depth review of these techniques is provided by Kumar and Raphael (2001).

When the set of attributes is fixed and when a single value is possible for each attribute, cases may be stored in a relational database. This is advantageous because relational database systems offer efficient information storage and retrieval.

Case-Based Reasoning System

This section describes a case-based reasoning system that helps identify good control solutions for the tensegrity structure. The following modules were implemented: Retrieval, Adaptation, Application, and Maintenance.

During retrieval, successfully solved control tasks are compared with the current task (see Fig. 10). A set of cases ranked by their degree of similarity is proposed for adaptation. After choosing the case to adapt, adaptation offers a choice of three stochastic search techniques to converge to a new control command. Use of the same stochastic search techniques to adapt cases was not envisaged in the original schema proposed by Shea et al. (2002), (Fig. 8). Maintenance is linked with retrieval of similar cases.

With a growing case base, the number of cases that require comparison with the current task increases and each similarity calculation slows case retrieval time. Comparing only relevant cases reduces the total number of comparisons, thereby increasing performance. The maintenance module groups cases in clusters, employing *k*-means clustering (Anderberg 1973). After calculating the distance between the current control task and the centroid of each cluster, only cases around the nearest cluster are considered during retrieval.

Similarity Measurement and Retrieval

The first phase in the use of a CBR system involves finding cases close to the current situation. Since the probability of retrieving an exact match is usually low, the solution part of a similar case has to be adapted. The challenge of developing procedures for similarity measurement includes:

- Identifying the case features/properties which are essential for similarity; and
- Selecting a similarity metric to be employed.

Sometimes case descriptions are mapped to numerical values for retrieval. In the scope of this work, such a mapping is not necessary since numerical values of the same type are being compared. Attributes chosen for comparison are as follows:

- Nodal displacements; and
- Strut positions.

An assumption of such retrieval is that a successful set of control commands can be reused for the same state and load case of the structure.

To avoid additional complexity, structural properties of the task description that have to correspond exactly to the stored task are as follows:

- Number and arrangement of tensegrity modules;
- Geometry of a tensegrity module;
- Location and type of supports; and
- Materials.

Only exact matches to the above attributes are proposed for case adaptation. As the exact detection of place and magnitude of loads might be too complex in practical situations, nodal displacement measurements are used to identify similar cases. Comparison of displacements employs the following "nearest neighbor" distance metric:

$$\text{distance}(X, Y) = \sqrt{\sum_{i=1}^d w_i^2 (x_i - y_i)^2} \quad (2)$$

where w_i = weight factor for the i th attribute set to 1 for all attributes in this study and x and y represent measured and stored nodal coordinates of a case.

For most applications, the number of measurements is limited. Selecting nodes to be used in the distance metric [Eq. (2)] involved simulating structural behavior for multiple load cases and choosing the nodes with the most significant overall displacements.

Selected nodes are identified by a circle. A verification process checked the metric. The comparison of telescopic bar positions always evaluates to "0" for the tests described in this paper since stored solutions have been obtained by starting the optimization from the initial "0" bar position.

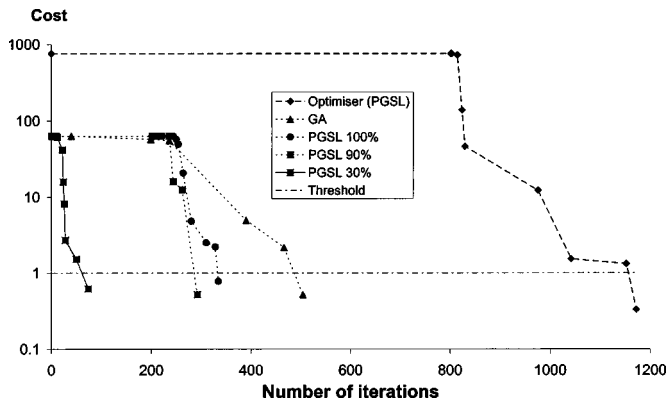


Fig. 11. Comparison of pure optimization and probabilistic global search Lausanne with different starting interval limitations

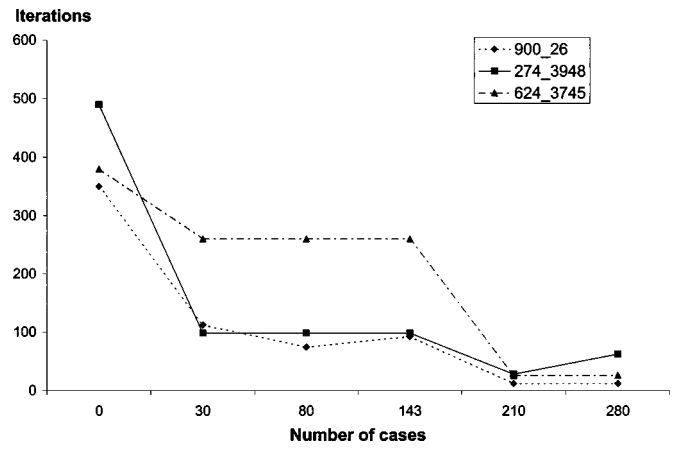


Fig. 12. Improvement of performance for three control tasks

Description of Test Case Base

To test the methodology on the structure and using the control setup, a reference case base was employed. Table 1 presents cases created for single node loading and Table 2 cases for loading on two nodes. Initial costs are also calculated with Eq. (1) for each task.

Solutions are found through optimization employing the PGSL algorithm. The threshold value was set to 1. This is a tradeoff between measurement accuracy, which has been evaluated to have a value of approximately 20 (Fest 2002) and the objective to have a sufficiently long iterative period of search for algorithm comparison. Three runs have been launched and each time the system produced three distinct solutions for each control task. Cells denoted in boldface in Tables 1 and 2 are control tasks that

have been evaluated on the structure. These cases will be reused for testing the CBR system; they have been chosen on the basis that initial costs should be ≥ 400 .

Evaluation of Case-Base Reasoning

Test Description

Tests on a three module structure (Domer 2003) with a similar control objective led to the decision to test the following techniques for case adaptation:

- PGSL and
- GAs.

Table 1. Single Node Loading Patterns with Initial Costs; for Node Numbers see Fig. 6

Load (N)	Nodes								
	6	12	16	26	32	37	39	41	48
150	55.73	8.52	19.97	103.30	106.18	159.28	242.53	2.02	163.63
200	74.43	11.73	26.62	137.08	141.48	212.63	323.28	2.82	218.38
274	102.18	15.77	36.37	186.58	193.38	291.78	442.63	4.17	299.43
300	112.03	17.32	39.82	203.73	211.58	319.53	484.53	4.72	327.98
350	130.98	20.22	46.32	236.68	246.48	373.18	565.18	5.77	382.88
391	146.48	22.72	51.77	263.53	275.13	417.23	631.00	6.67	427.93
450	168.93	26.27	59.47	301.73	316.13	480.78	726.28	8.22	492.98
508	191.18	29.82	67.07	338.88	356.33	543.38	819.73	9.72	556.93
550	207.28	32.32	72.57	365.53	385.33	588.78	887.43	10.87	603.33
625	236.08	36.92	82.27	412.73	436.88	670.08	1,008.38	13.02	686.33
650	245.78	38.52	85.62	428.23	454.08	697.23	1,048.58	13.87	714.08
700	265.18	41.57	92.07	459.28	488.38	751.53	1,129.48	15.52	769.48
742	277.05	44.27	97.52	485.03	517.03	797.38	1,197.53	16.87	816.13
800	303.63	47.77	104.97	520.38	596.53	860.68	1,291.63	18.88	880.63
859	325.98	51.52	112.62	412.68	596.55	928.23	1,387.73	21.09	976.46
900	341.73	54.17	117.87	580.48	624.23	970.28	1,454.63	22.65	992.28
976	370.83	59.07	127.67	625.53	675.53	1,053.83	1,579.03	25.76	1,077.18
1,000	380.08	60.57	130.72	639.58	691.68	1,080.23	1,618.38	26.72	1,104.13
1,050	399.33	63.72	137.07	668.83	725.18	1,135.38	1,700.58	28.83	1,160.18
1,092	414.13	66.37	142.47	693.18	753.30	1,192.93	1,786.34	31.14	1,286.68
1,209	461.24	73.97	157.37	760.18	831.13	1,311.58	1,963.60	36.01	1,339.01

Table 2. Loading Patterns on Two Nodes with Initial Costs; for Node Numbers see Fig. 6

Load (N)	Nodes						
	37 and 48	48 and 45	37 and 45	41 and 50	16 and 34	39 and 48	37 and 50
98	211.13	13.72	97.13	103.27	87.57	265.43	0.58
157	338.88	160.03	155.28	164.17	132.12	425.48	0.98
215	464.93	218.68	212.33	223.17	180.72	583.03	1.53
274	593.98	278.18	270.13	282.32	230.12	743.43	2.28
300	650.43	304.33	295.53	308.17	251.28	814.28	2.53
332	720.63	336.48	326.68	339.67	278.62	901.48	3.03
391	850.18	395.58	383.93	397.07	327.72	1,062.43	3.93
449	978.23	455.33	440.03	452.67	375.92	1,221.18	4.93
507	1,106.68	510.93	495.93	507.37	424.12	1,380.08	5.93
566	1,237.88	569.13	552.43	562.32	472.92	1,542.73	7.18
624	1,367.38	626.23	607.78	615.37	520.72	1,703.33	8.48
650	1,425.53	651.63	632.38	638.92	524.27	1,775.63	9.18
700	1,537.73	700.48	679.78	683.67	583.32	1,915.08	10.43

Since each adaptation process has been carried out three times; the results of the best solutions have been plotted. Four test series are described as follows:

1. **The potential of CBR to improve performance.** This test is similar to those made on the three-module structure. Performance of “pure” optimization using PGSL is compared with CBR using PGSL and GAs for adaptation.
2. **Genetic algorithms for “pure” optimization as well as GAs for adaptation.** In Series 1, the impact of using GAs to adapt cases from a similar task is compared with pure optimization using PGSL. This is a comparison of adaptation results with pure optimization employing GAs.
3. **Clustering to assist case maintenance.** The way cases are organized in a case-base influences efficacy of retrieval. *K*-means clustering is tested as a methodology to decrease retrieval time.
4. **Performance enhancement over time.** By adding good cases to the case base, adaptation performance is expected to increase. The influence of case-base size is also studied.

Series 1: Potential of Case-Base Reasoning for Performance Improvement

Best-so-far curves are used to compare performance. In general, adaptation employing GAs converged faster than pure optimization. The PGSL performed better than any other adaptation technique in every tested case.

A representative comparison of pure optimization with optimization employing different starting interval limitations is given in Fig. 11. Starting interval limitations are the allowable bounds on variable values when search starts from a previous solution (case). 100% is the variable range that would be used in a search task that did not employ a case. The GA does worse than all PGSL options when adapting cases.

Series 2: Genetic Algorithms for All Search Tasks

Although the GAs converge faster in some situations, this does not indicate that they performed better at adaptation than pure optimization involving GAs. In many cases, the number of iterations needed is observed to be the same for adaptation and pure optimization. The same behavior is observed for some tasks. For

other tasks, adaptation performed even worse than pure optimization. This indicates that GAs are not as advantageous for adaptation as PGSL.

Series 3: Testing Clustering to Assist Case Maintenance

Preliminary tests indicated that retrieval and thereby the process of calculating case distances might outweigh computational time needed for adaptation. Clustering cases to limit the number of cases to be examined during retrieval is proposed to speed up this process. Different numbers of clusters have been evaluated on the set of cases present in the reference case base. As quality criterion to identify good clustering, the Calinski–Harabasz (CH) criterion has been employed (Legendre 2001)

$$CH = \frac{[R^2/(K-1)]}{[(1-R^2)/(n-K)]} \quad (3)$$

$$R^2 = \frac{SST - SSE}{SST}$$

where SST=total sum of squared distances to the overall centroid and SSE=sum of squared distances of the objects to their own centroids.

The criterion has to be calculated for different values of *K*. Large values indicate a good clustering. The three most promising ones, *K*=5, 35, and 100, have been tested for cases in boldface in Tables 1 and 2. Results are given in Table 3. Control tasks are identified by the magnitude of load and the nodes where the load is attached. For example, control task “215_39&48” means that a load of 215 N acts on Nodes 39 and 48.

Time decreased significantly for *K*=5. Retrieved cases are identical to those retrieved for *K*=0. More precisely: without decreasing retrieval quality, clustering resulted in a speed up by a factor of 3. With a value of *K*=35, results are still excellent: in most cases, at least the first one or two cases proposed were still the same. A supplementary speed up by a factor of 3 has been obtained. For *K*=100, retrieval quality decreased further without increases in speed.

Table 3. Results for Different Values of k

Control task	Retrieved cases							
	$k=0$		$k=5$		$k=35$		$k=100$	
	Case	Time (min:s)	Case	Time (min:s)	Case	Time (min:s)	Case	Time (min:s)
391_48	350_48	02:18	350_48	00:44	350_48	00:14	450_48	00:12
	450_48		450_48		450_48		215_37&48	
	215_37&48		215_37&48		215_37&48			
	508_32		508_32		508_32			
	157_37&48		157_37&48		550_32			
550_48	508_48	02:19	508_48	00:44	274_37&48	00:10	508_48	00:11
	274_37&48		274_37&48		300_37&48			
	625_48		625_48		742_32			
	300_37&48		300_37&48		700_32			
	450_48		450_48		508_37			
625_26	650_26	02:18	650_26	00:49	650_26	00:09	650_26	00:11
	700_26		700_26		550_26			
	550_26		550_26		508_26			
	742_26		742_26		450_26			
	508_26		508_26		391_26			
215_39&48	300_39	02:23	300_39	00:49	300_39	00:14	-	00:11
	350_39		350_39		274_39			
	274_39		274_39		157_39&48			
	157_39&48		157_39&48		508_48			
	274_39&48		274_39&48		450_37			
274_39&48	300_39&48	02:23	300_39&48	00:50	300_39&48	00:09	300_39&48	00:13
	391_39		391_39		391_39		391_39	
	332_39&48		332_39&48		332_39&48		450_39	
	215_39&48		215_39&48		450_39		350_39	
	450_39		450_39		350_39			

Series 4: Performance Enhancement Over Time

Compared with pure optimization, performance increases are observed when adapting good solutions close to the current task. In practical situations, these solutions are added to the case base after validation on the structure. By constantly adding good cases, the case-base reasoning system is able to improve performance over time.

A growing cases base is simulated through studying the performance of the system using different case-base sizes. Case-Base I might represent a system at the beginning of its learning process, whereas Case-Base V could be close to a system that has already solved multiple cases and can be considered to be "more mature" (Table 4).

Case-base sizes inbetween have been obtained by deleting four cases in each step while descending from Case-Base V to Case-Base I. The PGSL has been used for all adaptation processes. Results are plotted in graphs that compare the different stages of the growing case base and pure optimization regarding iterations needed to attain a best state. When increasing case-base size results in fewer iterations the structure learns from its experience. Examples of this behavior are shown in Fig. 12. In some situa-

tions, performance decreased slightly when using a bigger case base. Decreases are not significant; however, they can be related to the stochastic nature of the adaptation process.

Conclusions

Although they converged faster than PGSL during pure optimization, GAs do not perform best when launched close to a good solution. Stable iterative performance observed in conjunction with tests on the three-module structure indicates that GAs perform better than PGSL during pure optimization. Seeding multiple good solutions into the initial population might increase performance during adaptation.

The presence of multiple search algorithms for adaptation and optimization is justified, since techniques are task dependent and relative performance might change with other control objectives.

For case storage, priority should be given to cases that can easily be adapted and fit well into the space of solutions needed for anticipated tasks. The distance metric helped to retrieve good cases for adaptation, but did not retrieve the optimal case for adaptation in every case. This aspect requires further study.

Although the case base for the five-module structure contains many entries, retrieval time of cases is less than adaptation time. Clustering shows the potential to speed up case retrieval without affecting the system's competence. The interesting aspect of clustering is that the same distance metric that is used for retrieval can

Table 4. Test Case Bases

Case base	I	II	III	IV	V
Number of cases	30	80	143	210	280

be applied. Performance decreases observed with $k=100$ can be explained with the time the clustering algorithm needs to find the closest centroid. Competence reduction is related to the size of the case base (280 cases). Only a small number of cases are in each cluster. Virtual centroids might be seen as pivotal cases in the classification of Smyth and Keane (1995). Clustering itself creates an additional administrative task for the case base: recluster when adding new cases may not be convenient due to the computation time that is necessary.

Tests employing different case-base sizes showed that the system improves performance over time with growing case bases. Therefore, the structure learns from its experience. Increases with the last two case bases (IV and V) are not as important as in the beginning with Case bases I–III. This indicates that the case-base size of 280 cases covers the space of possible solutions sufficiently for the range of load cases under study. Adding further cases will result in only small performance enhancements. Enhancements of the case base regarding other load cases including horizontal loads as well as other control objectives are thus possible without excessive case-base size.

Acknowledgments

The writers would like to thank the Swiss National Science Foundation for supporting this work under Contract No. 21-54117.98. They also thank Dr. Etienne Fest who constructed the actively controlled full-scale tensegrity structure and designed the connections, Dr. Benny Raphael for providing support during programming the intelligent control system, and Dr. Kristina Shea for help in launching the project, Intelligent Tensegrity Structures, with Professor Ian. F. C. Smith. Contributions of student projects by Antje Landschulz, Vikram Lalit, Gaurav Gupta, and Lukas Bieri are also recognized.

References

- Anderberg, M. R. (1973). "Nonhierarchical clustering methods." *Cluster analysis for application*, Academic, New York, 156–175.
- Averseng, J., Kazi-Aoual, M. N., and Crosnier, B. (2002). "Tensegrity systems selfstress state implementation methodology." *Proc., 5th Int.-Conf. on Space Structures*, University of Surrey Press, Guildford, U.K., 31–38.
- Barnes, M. R. (1977). "Form finding and analysis of tension space structures by dynamic relaxation." PhD thesis, The City Univ. of London, London.
- Connor, A. M., and Shea, K. (2000). "A comparison of semi-deterministic and stochastic search techniques." *Evolutionary design and manufacture, selected papers from ACDM '00*, Univ. of Plymouth, Plymouth, U.K., 287–298.
- Djouadi, S., Motro, R., Pons, J. C., and Crosnier, B. (1998). "Active control of tensegrity systems." *J. Aerosp. Eng.*, 11(2), 37–44.
- Domer, B. (2003). "Performance enhancement of active structures during service lives." PhD thesis, Swiss Federal Institute of Technology, Lausanne, Switzerland, (<http://library.epfl.ch/theses/>), (April 2004).
- Domer, B., Fest, E., Lalit, V., and Smith, I. F. C. (2003). "Combining dynamic relaxation method with artificial neural networks to enhance simulation of tensegrity structures." *J. Struct. Eng.* 129(5), 672–681.
- Domer, B., Raphael, B., Shea, K., and Smith, I. F. C. (2003). "A study of two stochastic search methods for structural control." *J. Comput. Civ. Eng.*, 17(3), 132–141.
- Dowland, K. (1995). "Simulated annealing." *Modern heuristic techniques for combinatorial problems*, C. Reeves, ed., McGraw-Hill, London, 20–69.
- El-Beltagy, M. A., and Keane, A. J. (1999). "A comparison of various optimization algorithms on a multilevel problem." *Eng. Applic. Artif. Intell.*, 12, 639–654.
- Fest, E. (2002). "Une structure active de type tensegrité." PhD thesis, Swiss Federal Institute of Technology, Lausanne, Switzerland, (<http://library.epfl.ch/theses/>), (April 2004).
- Fest, E., Shea, K., Domer, B., and Smith, I. F. C. (2003). "Adjustable tensegrity structures." *J. Struct. Eng.*, 129(4), 515–526.
- Goldberg, D. E. (1989). *Genetic algorithms for search, optimization and machine learning*, Addison-Wesley, Reading, Mass.
- Kahla, N. B., Moussa, B., and Pons, J. C. (2000). "Nonlinear dynamic analysis of tensegrity systems." *J. Int. Assoc. Shell Spatial Structures: IASS*, 41(132), 49–58.
- Kolodner, J. (1993). *Case-based reasoning*, Morgan Kaufmann, San Mateo, Calif.
- Kumar, B., and Raphael, B. (2001). *Derivational analogy based structural design*, Saxe Coburg, Stirling, U.K.
- Leake, D. B. (1996). *Case-based reasoning: Experiences, lessons, & future directions*, D. B. Leake, ed., California Press, Menlo Park., Calif.
- Leake, D. B. (2001). "Artificial intelligence." *Van Nostrand Scientific Encyclopedia*, Wiley, New York.
- Legendre, P. (2001). *Program K-means User's Guide*, University of Montreal Press, Montréal, Canada, (www.fas.umontreal.ca/biol/casgrain/en/labo/k-means.html) (April 2004).
- Manoharan, S., and Shanmuganathan, S. (1999). "A comparison of search mechanisms for structural optimization." *Comput. Struct.*, 73(1–5), 363–372.
- Motro, R. (1992). "Tensegrity systems: The state of the art." *Int. J. Space Struct.*, 7(2), 75–83.
- Motro, R., and Raducanu, V. (2001). "Tensegrity systems and tensile structures." *Proc., Int. Symp. on Theory, Design and Realization of Shell and Spatial Structures* (CD Rom), Nagoya, Japan.
- Murakami, H. (2001a). "Static and dynamic analysis of tensegrity structures. Part I. Nonlinear equations of motion." *Int. J. Solids Struct.*, 38(20), 3599–3613.
- Murakami, H. (2001b). "Static and dynamic analysis of tensegrity structures. Part II. Quasistatic analysis." *Int. J. Solids Struct.*, 38(20), 3615–3629.
- Oppenheim, I. J., and Williams, W. O. (2001). "Vibration and damping in three-bar tensegrity structure." *J. Aerosp. Eng.*, 14(3), 85–91.
- Raphael, B., and Smith, I. F. C. (2003). "Direct stochastic algorithm for global search." *Appl. Math. Comput.*, 146(2-3), 729–758.
- Salama, M., Umland, J., Bruno, R., and Garba, J. (1993). "Shape adjustment of precision truss structures: Analytical and experimental validation." *Smart Mater. Struct.*, 2(4), 240–248.
- Shea, K., Fest, E., and Smith, I. F. C. (2002). "Developing intelligent tensegrity structures with stochastic search." *Adv. Eng. Inf.*, 16(1), 21–40.
- Skelton, R. E., et al. (2000). "An introduction to the mechanics of tensegrity structures." *Handbook on mechanical systems design*, CRC, Boca Raton, Fla.
- Smyth, B., and Keane, M. T. (1995). "Remembering to forget." *Proc., 14th Int. Joint Conf. in Artificial Intelligence*, Montréal, 377–382.
- Sultan, C. (1999). "Modeling, design, and control of tensegrity structures with applications." PhD thesis, Purdue Univ., West Lafayette, Ind.
- Sultan, C., Corless, M., and Skelton, R. E. (2002). "Linear dynamics of tensegrity structures." *Eng. Struct.*, 24(6), 671–685.
- Williamson, D., and Skelton, R. E. (1998). "A general class of tensegrity systems." *Engineering mechanics for the 21st century*, La Jolla, Calif.
- Wolpert, D. H., and Macready, W. G. (1997). "No free lunch theorems for optimization." *IEEE Trans. Evolutionary Computation*, 1(1), 67–82.