An Evolutionary Algorithm for the Block Stacking Problem

Tim Hohm, Matthias Egli, Samuel Gaehwiler, Stefan Bleuler, Jonathan Feller, Damian Frick, Richard Huber, Mathias Karlsson, Reto Lingenhag, Thomas Ruetimann, Tom Sasse, Thomas Steiner, Janine Stocker, and Eckart Zitzler*

Computer Engineering and Networks Laboratory (TIK), ETH Zurich
{hohm,bleuler,zitzler}@tik.ee.ethz.ch,
http://www.tik.ee.ethz.ch/sop/

Abstract. How has a stack of n blocks to be arranged in order to maximize its overhang over a table edge while being stable? This question can be seen as an example application for applied statics and at the same time leads to a challenging optimization problem that was discussed recently in two theoretical studies.

Here, we address this problem by designing an evolutionary algorithm; the proposed method is applied to two instances of the block stacking problem, maximizing the overhang for 20 and 50 block stacks. The study demonstrates that the stacking problem is worthwhile to be investigated in the context of randomized search algorithms: it represents an abstract, but still demanding instance of many real-world applications. Furthermore, the proposed algorithm may become useful in empirically testing the tightness of theoretical upper bounds proposed for this problem.

1 Introduction

What is the largest overhang beyond the edge of a table that can be reached with a stack of n blocks (see Fig. 1)? A question that can be reformulated into the following optimization problem: Find a stable stack consisting of n blocks with a maximal overhang beyond the edge of a table. This is an example system to demonstrate the principles of static equilibrium that was discussed in two recent theoretical studies by Hall [3] and Paterson and Zwick [6].

Although this problem is mentioned in several engineering mechanics textbooks throughout the years [7,4] as well as in mathematics [1,5], until now optimal solutions are only known for strongly restricted variants. Nevertheless construction schemes and upper bounds for some scenarios are available [3,6]. Up to now, there exists no algorithm generating stacks where each block is explicitly represented; both studies, Hall's and Paterson and Zwick's, present algorithms involving implicit representations for parts of the stacks by introducing additional weights.

* This study arose from a student project within the Bachelor's program in Information Technology and Electrical Engineering at ETH Zurich that was supervised by Stefan Bleuler, Tim Hohm and Eckart Zitzler.

N. Monmarché et al. (Eds.): EA 2007, LNCS 4926, pp. 111–122, 2008.
© Springer-Verlag Berlin Heidelberg 2008

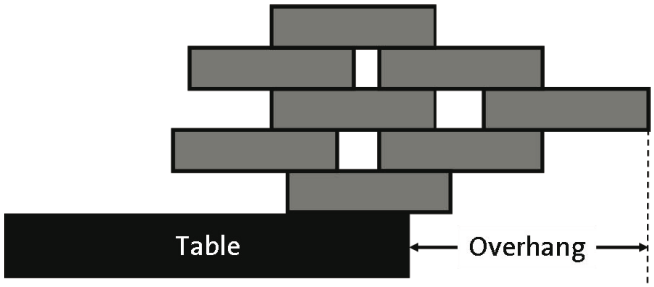

Fig. 1. An example Stack and its overhang over a table edge

Therefore developing a technique generating fully represented stack provides an interesting complement to the theoretical studies, empirically substantiating the proposed bounds. In addition, following Hall, the described problem *" ... could pose a worthy test for general optimization algorithms."* (page 1115 in [3]). This problem is an interesting addition to the set of test problems for stochastic optimization techniques like evolutionary algorithms (EAs) and expands the set of representatives of stacking problems often considered in EA studies [2].

Therefore, taking up Hall's suggestion in the following, we describe an EA designed to work on the described problem, addressing the questions of (i) how to represent the problem and how to design appropriate variation operators such that an EA can explore the space of stack configurations effectively, and (ii) what overhang can be reached by an EA - regarding a 20 block and a 50 block setup.

2 Related Work

The block stacking problem has a long history and was mentioned already in the 19th century [8] recurring from then on regularly textbooks, providing only limited information on how a general optimal solution could look like. Recently, two theoretical studies examined the problem in more detail. In the first study, Hall [3] investigated the influence of different restrictions of the problem on the achievable overhang. He showed that the widely believed to be optimal overhang D_n for a stack of n blocks of

$$D_n = \frac{1}{2} \sum_{i=1}^{n} \frac{1}{i} \tag{1}$$

only holds for the most restricted variant: a scenario considering only vertical forces on the blocks and further demanding that all blocks are lying one-on-one. Non-intuitively, since Eq. 1 diverges for $n \to \infty$, the overhang for stacks following these restrictions can already reach an infinite overhang. Additionally he tested two less restricted variants, one where as well only vertical forces are considered but more than one block is allowed to rest on top of another and one extending the former scenario by considering friction as vertical contribution. For both less restricted settings better overhangs could be achieved.

Taking up the work by Hall and focusing on stacks of where more than one block is allowed per level and only vertical forces are considered, in a succession study Paterson and Zwick [6] further formalized the problem and introduced upper bounds, whereas their tightest bounds remain unproven. Further on they propose construction schemes for a slightly varied problem where they only consider a set of blocks producing the actual overhang and the remaining blocks are only implicitly represented by point forces applied from above. Using this variated problem (loaded stacks), empirical results for different numbers of blocks are presented.

Thereby these two studies provide formalizations and theoretical background for the considered problem, paving the way to consider the described problem as a demanding test problem for general optimization techniques. In addition, the tightest proposed bounds for optimal overhangs are given for loaded stacks only therefore posing the questions what an algorithm designing stacks with optimal overhang looks like and if the bounds can be reached.

3 The Block Stacking Problem

The problem considered here is to find a stack of n blocks producing a maximal overhang over the edge of a table while being stable, see Fig. 1. In this study the two dimensional block stacking problem is investigated for which it is assumed that:

- all blocks are rectangular,
- all blocks are of same size,
- all blocks are rigid,
- all blocks are perfectly smooth (no friction between blocks),
- all blocks are of equal and same density,
- all blocks have to lie on their long edge.

Further on, it is assumed that the table on which the stacks are built covers the third quadrant ($x, y \leq 0$) of a Cartesian coordinate system in which the origin marks the upper right corner of the table.

In a stack built from such blocks, contacting blocks exert forces onto each other. These forces can be summed up to a single resulting force acting at one point in the contact interval. Resulting from the assumption that there is no friction between blocks, no horizontal forces but only vertical forces are present and since their is no drag between blocks, all forces F exerted have to be non-negative ($F \geq 0$). Following Newton's third law of motion, for each force F_A exerted by a block A on a block B, there has to be a counterforce $F_B = -F_A$ exerted from block B onto block A.

Now, for a stack to be stable it is necessary that all blocks in this stack are in equilibrium—a condition met if all forces exerted by blocks lying on top of a considered block C (plus the weight force F_W of C) and the moments imposed by these forces are evened out by counter forces and moments exerted by blocks lying below C. For the example stack shown in Fig. 2, for the middle light gray

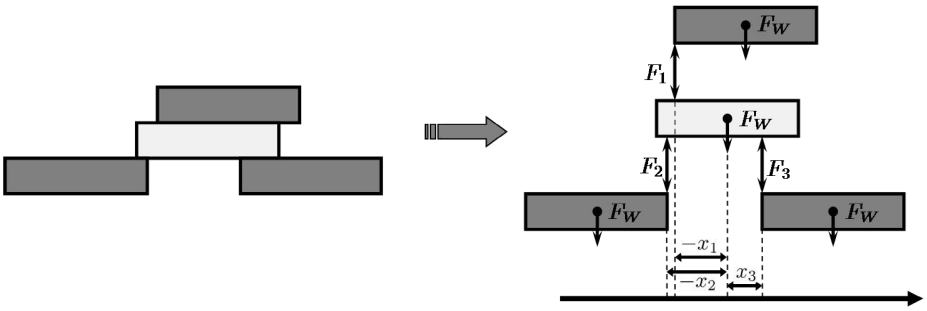

Fig. 2. To determine the forces exerted on or by a block A, a stack like the one shown on the left is decomposed into single blocks (shown on the right). Adhering gravity, each of these blocks has a downwards directed weight force F_W. In addition, there are forces exerted between the blocks, summed up to resulting forces that act on a certain point of the contact surface and for each of these forces exists an exact counterforce (F_1, F_2, F_3). Finally, momenta act on each block. They are defined by the forces acting on the block times the horizontal parts of the vectors between the forces' points of application and the block's centroid.

block this results in the following equations, Eq. 2 and Eq. 3, that need to be satisfied:

$$F_1 + F_W = F_2 + F_3 \tag{2}$$

$$x_1 F_1 + x_3 F_3 = x_2 F_2 \ , \tag{3}$$

where x_i denotes the horizontal part of the distance of the point where F_i is exerted to the centroid of the block. If and only if there exists a set of non-negative forces for which these equations for all blocks in the stack are satisfied, the stack is stable.

The model described here is the same that was already used in the studies of Hall and Paterson and Zwick, representing an abstraction of the full three dimensional block stacking problem that, i.e., can be tested by using wooden blocks like in the game *J*enga and allows for fast fitness function evaluations due to its simplicity. Additionally, it can be easily extended by, i.e., incorporating friction between blocks or using non-identical block shapes. Still, the principles underlying the optimization will stay the same, only the question if a stack is stable or not will become harder to answer.

4 An EA for the Block Stacking Problem

In the following, an EA for the block stacking problem is proposed, in particular (i) the choice of a suitable representation, (ii) evolutionary operators for generating offspring, and (iii) the considered fitness function are described. Here, choosing a suitable stack representation poses the most important but most difficult task since straight forward representations often suffer from resulting in

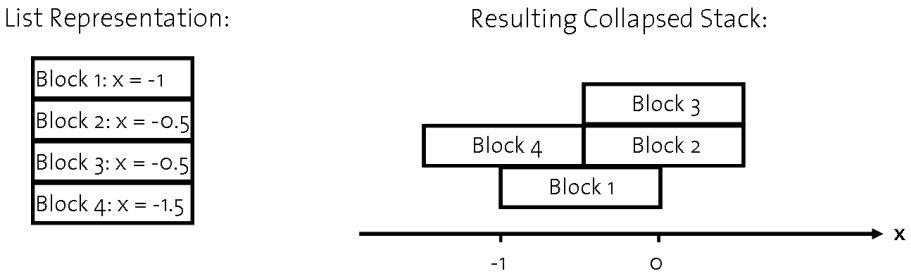

Fig. 3. Schematic overview on the transition process from representation to a stack

physically invalid stacks due to overlapping blocks and thereby make it hard to design functioning evolutionary operators.

4.1 Representation and Candidate Stack Initialization

How to represent a stack in a way that no two blocks overlap is less obvious then it appears. For example, when representing the stack by storing the x, y coordinates of a certain point of each block, without using a repair mechanism overlapping blocks can occur. To avoid this problem we have decided to use a representation which by default produces feasible stacks: A stack s is represented by an ordered list $s = [b_1, \ldots, b_n]$ of blocks where for each block b_i the x coordinate of its lower left corner is stored. To generate the stack represented by this list, the blocks are dropped from above onto the table one-by-one according to their order given in the list. Their y coordinate is then determined by the level down to which they fell before hitting either the table or another block (cf. Fig. 3). Further on, since only stacks with just a single block lying directly on the table can have an optimal overhang, stacks have to fulfill this criterion to be valid. If invalid stacks are occurring during optimization or initialization, the process leading to the invalid stack is repeated until a valid stack is generated. In addition to this validity condition, during initialization of the first individuals further criteria are posed on the stacks that need to be fulfilled. For initialization, x coordinates are randomly drawn from a normal distribution $\mathcal{N}(-1, \sigma)$[1], where σ is iteratively increased as the stack grows from bottom to top: starting from $\sigma = 0$ with each new x coordinate drawn for a stack (one block is added), σ is increased by a constant s. For a stack to become valid, it is then required that each of the newly placed blocks has a minimum contact surface to blocks on top of which they are placed. For the first block the minimal overlap in percent o is defined by an offset constant o_{off}, iteratively increased with each block so that for the last block a predefined overlap value o_{max} is reached. The iterative

[1] Since each block is represented by the x coordinate of its lower left corner and each block has a breadth of 1, the normal distribution the x coordinates are drawn from has a mean of -1: this allows for generation of stacks where the first block in expectation comes to rest with its centroid above the table.

increase for o is calculated by linearly distributing $o_{max} - o_{off}$ on the n blocks of the stack. Using this method, a hundred times the number of required stacks are generated and those stacks with the minimal stack height are chosen.

4.2 Operators

The proposed algorithm involves two selection steps, mating selection and environmental selection as well as mutation and crossover. For mating selection a tournament selection with tournament size two is used. Taking thereby selected pairs of individuals, they are first recombined and one of the resulting offspring individuals is afterwards mutated. Recombination takes place according to a predefined crossover probability p_{cross} and in case no recombination is applied, simply the first parent is handed over to mutation. After mutation, a plus-selection scheme on parent population and offsprings is used to determine the new population by taking the best individuals from this set.

Crossover and mutation are working in detail as follows: For recombination, one point crossover is used, choosing the crossover point P_{cross} uniformly distributed $P_{cross} \in \{1, 2, \ldots, n\}$, where n is the number of blocks in the stack. Afterwards the resulting offspring stack is collapsed to check if it adheres to the validity constraint (only one block lies on the table, none is falling in the void beyond the table). If the resulting stack is invalid, crossover is repeated up to 99 times and if within these 100 tries no valid stack is produced, crossover is omitted.

For mutation, one of the blocks in the stack is chosen uniformly random. This block and all the blocks on top of it in the collapsed representation are then moved according to a random movement drawn from a normal distribution $\mathcal{N}(0, \sigma)$, where σ is a predefined mutation strength. If this movement results in an invalid stack, mutation is repeated up to 99 times and if within these 100 tries no valid stack is created, the parent remains unchanged and is added to the set of offsprings. By using this mutation, stacks that are loosing height are only scarcely created although in Sec. 2 it was shown that it is necessary to have stacks with a height smaller than their number of blocks to reach optimal overhang. Therefore, in one percent of all mutation cases a different type of mutation is used: only the x coordinate of the chosen block is changed while all the others stay the same. This mutation variant is more likely to produce stacks loosing height.

4.3 Fitness Evaluation

The aim of the optimization process is to identify stable stacks with a maximal overhang, therefore to evaluate a given stack two questions need to be answered: first, is the considered stack stable and second, what is its overhang.

For a stack to be stable, in Sec. 3 it was shown that all blocks need to be in equilibrium. Therefore, for each block the corresponding equations for forces and resulting moments under the constraint that forces between blocks are non-negative and that weight forces are strictly positive, have to be set up. If and only if there exists a feasible force distribution for this usually under-determinated

problem, the tested stack is stable. The question if there exists a feasible solution can be formulated as a linear programming problem that in turn can be solved using standard solvers like Matlab, an operation consuming about three milliseconds for a 20 block stack on one core of a Dual Core Double CPU AMD Opteron 2.6GHz 64 Bit machine with 8GB RAM.

The overhang on the other hand can be calculated by determining the position of the rightmost block boundary or the greatest x-extension of a block—the table was located in the third quadrant and therefore the table edge is located at the origin of the Cartesian coordinate system fitted into the two dimensional space.

Since we deemed it easier for the optimization process to improve the overhang starting from a stable stack than stabilizing an instable stack with good overhang, we decided to use as a fitness $f(x)$ of stack x its overhang $over(x)$ if the stack is stable and zero otherwise which is given by the following equation:

$$f(x) = \begin{cases} over(x) & \text{if x stable} \\ 0 & \text{else.} \end{cases} \tag{4}$$

Thereby stable stacks are always preferred compared to instable ones.

5 Simulation Results

The proposed approach is tested by applying it to two instances of the block stacking problem, one using 20 block stacks and the other one stacks made up from 50 blocks. While the latter case represents a problem size that is on the verge of becoming inaccessible to exhaustive search procedures and thereby provides a glimpse on the general applicability of the proposed method, the first is used mostly for parameter optimization for the latter case. In the following first the results of the parameter optimization are presented, followed by the simulation results on the 50 block stack optimization.

5.1 Parameter Testing

During the parameter optimization process we tested mutation- and crossover rates, population- and offspring set sizes and different settings for the initialization method.

Starting with the initialization technique, offset values $o_{off} \in \{10, 20, 30, 40, 45\}$ and maximal overlaps $o_{max} \in \{50, 60, 70, 80, 90, 99\}$ were tested. Hereby, the aim was to identify a good trade-off between stack stability and stacks with small height: initial simulations showed that the optimization process becomes difficult if only few stable stacks are in the initial population (data not shown) which is often the case if only small overlaps for the blocks are required. On the other hand, to reach a good overhang it is necessary to build stacks of small height, allowing for counterweight stacks to emerge (cf. Sec. 2). Therefore we tested 10000 stacks for each combination of o_{off} and o_{max}, counting the number of stable stacks (cf. Tab. 1) with the variant $o_{off} = 45$ and $o_{max} = 99$ building the

Table 1. Trade-off between overlaps between blocks and stability measured for 10000 randomly chosen individuals, for 20 and 50 block scenarios

o_{off}	o_{max}	Fraction of stable stacks for 20 block stacks	Fraction of stable stacks for 50 block stacks
10	50	0.0003	0.0000
10	60	0.0016	0.0000
10	70	0.0052	0.0000
10	80	0.0135	0.0000
10	90	0.0275	0.0003
10	99	0.0415	0.0003
20	50	0.0017	0.0000
20	60	0.0073	0.0000
20	70	0.0192	0.0002
20	80	0.0340	0.0003
20	90	0.0618	0.0007
20	99	0.0874	0.0013
30	50	0.0070	0.0001
30	60	0.0218	0.0000
30	70	0.0434	0.0003
30	80	0.0719	0.0021
30	90	0.1071	0.0049
30	99	0.1471	0.0067
40	50	0.0205	0.0001
40	60	0.0364	0.0005
40	70	0.0746	0.0018
40	80	0.1185	0.0053
40	90	0.1793	0.0089
40	99	0.2378	0.0143
45	50	0.0251	0.0001
45	60	0.0474	0.0006
45	70	0.0912	0.0017
45	80	0.1493	0.0021
45	90	0.2285	0.0083
45	99	0.3107	0.0185

best compromise between stability and stack height. In turn, when testing the initialization method for 50 block stacks, the fraction of stable stacks dropped considerably to a value of 0.0185 still feasible for optimization but indicating that for even larger stacks the initialization technique might fail. To address this problem there are at least two possibilities, (1) couple the initialization technique with a local search technique trying to modify the stacks in such a way that they become stable or (2) initialize the stacks by using a set of stack designs that are stable by default.

Further on the influence of mutation strength and crossover rate were investigated. For mutation, different percentage levels $\sigma_{mut} \in \{0, 12.5, 25, 27.5, 50\}$ with respect to the unit block breadth have been tested while for recombination, probabilities $p_{cross} \in \{0.0, 0.25, 0.5, 0.75, 1\}$ have been tested. To measure

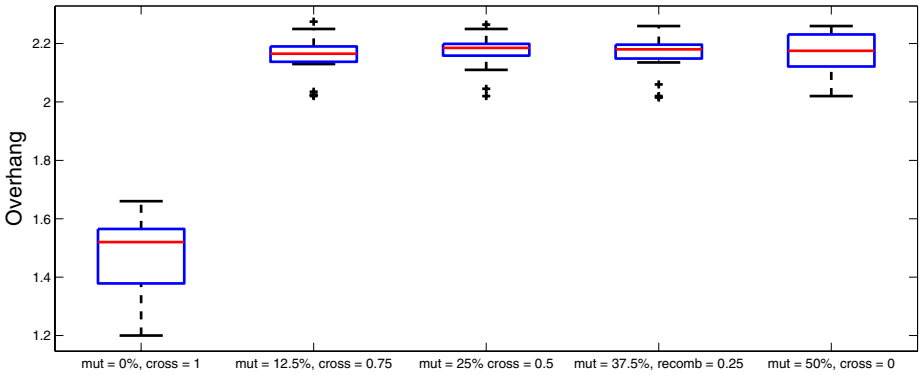

Fig. 4. Factorial design results for different mutation strengths (mut) and recombination probabilities (cross)

the overall performance, 21 optimization runs with a population size $\mu = 100$ and a offspring set size $\lambda = 200$ and 100000 objective function evaluations have been conducted. The results are shown in Fig. 4, indicating that mutation is necessary for the optimization process but as soon as mutation takes place (even if only slight movements of blocks are made) the overall performance for different mutation strengths is comparable. Only the setting where no recombination takes place shows a discernible loss of performance. For the further simulations we chose a mutation strength of $\sigma_{mut} = 25\%$ and a crossover probability of $p_{cross} = 0.25$. Using the determined parameters for initialization, mutation strength and recombination rates, in a last step we tested different population sizes μ and offspring set sizes λ. The tested settings have been $(\mu, \lambda) \in \{(10, 20), (100, 200), (200, 400)\}$. Again for each setup 21 runs, running for 100000 objective function evaluations, where conducted. The simulations showed that the mean performance for all three settings is comparable (cf. Fig. 5), whereas the variance for $(\mu, \lambda) = (10, 20)$ is much higher than for the latter two settings. In turn there is no reduction in variance when moving from $(\mu, \lambda) = (100, 200)$ to $(\mu, \lambda) = (200, 400)$ while the number of evaluations needed to reach a good overhang was smaller for $(\mu, \lambda) = (100, 200)$ than for $(\mu, \lambda) = (200, 400)$ (cf. Fig. 5.1), indicating that the population size of $(\mu, \lambda) = (200, 400)$ already could be a bit to large. In effect the best set of parameters identified during parameter optimization comprises the following settings: $o_{off} = 45$ and $o_{max} = 99$ for initialization, $\sigma = 25\%$ as mutation strength, $p_{cross} = 0.25$ as recombination probability and $(\mu, \lambda) = (100, 200)$ as population size or offspring set size respectively.

5.2 Application to a 50 Block Problem

The stacks evolved during the parameter optimization process already showed good overhangs—the best 20 block stack found had an overhang of 2.275 while for this stack size an overhang of 2.32014 is optimal and for a 19 block stack the

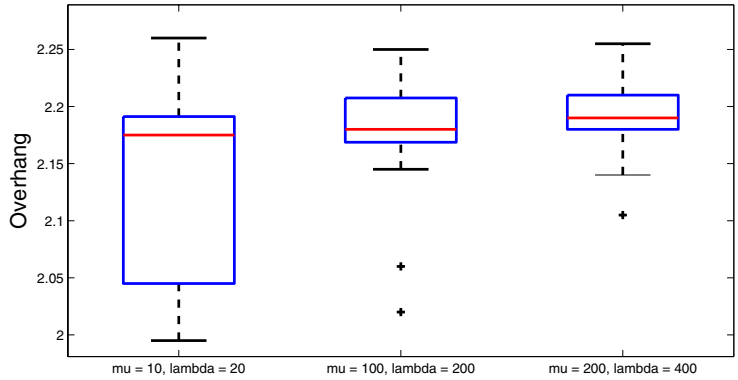

Fig. 5. Overview on the distributions of the best individuals found during 21 runs for different population sizes μ and offspring sizes λ

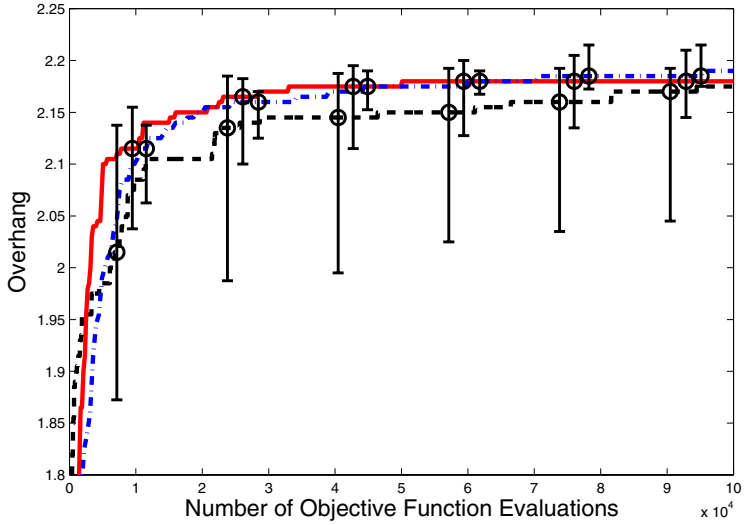

Fig. 6. Evolution of the best fitness per generation averaged over 21 runs for different population sizes. The error bars indicate the first and third quartile over the 21 runs. The dashed line represents a (μ, λ) combination of $(10, 20)$ (left error bar), the solid line of $(100, 200)$ (middle error bar) and the dash-dotted line of $(200, 400)$ (right error bar).

optimal overhang is 2.27713 respectively. Both optima have been determined in [6] by exhaustive search. A simple exhaustive search results in a runtime which is exponential in the number of used blocks. However, the problem complexity itself has not been determined yet. In addition to the good overhangs generated during parameter testing, the stack construction shown in Fig. 7 closely resembles the construction of the optimal ones given by Paterson and Zwick (cf. [6], Fig. 3, page 233). Taking this as a promising prospect, we decided to further test our

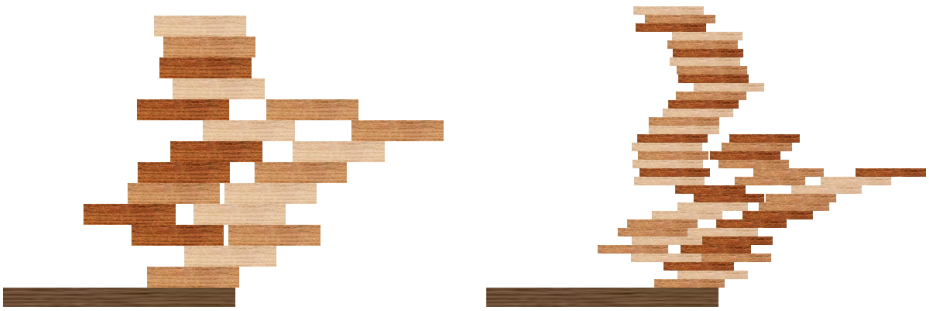

Fig. 7. Best stacks identified, for 20 blocks on the left (overhang = 2.275) and for 50 blocks on the right (overhang = 2.97)

approach on a more demanding instance of the problem: stacks containing 50 blocks. Conducting 50 runs, 100000 fitness function evaluations each, a stack with overhang 2.97 was found (cf. Fig. 7). According to Paterson and Zwick, a relatively tight putative upper bound for this problem instance is an overhang of 3.28136 and for 40 block stacks of 3.02248 [6]. Therefore the best stack found during optimization represents a solution with some distance to the optimum, still it is a good solution serving as a proof of principle that the proposed EA is in general capable to optimize the given stack overhang problem. Nevertheless the algorithm can be improved, first by introducing a method trying to repair instable stacks by systematically introducing small changes looking for stable stacks in the neighborhood of a given solution and second by using a type of local search trying to find the stack with the best overhang in direct vicinity to the given stack.

6 Conclusions

We have presented a preliminary study, originating from a student project, concerned with the optimization of stack overhangs over a the edge of a table, an example problem for static equilibrium as well as an interesting test problem for combinatorial optimization methods. The proposed algorithm was applied to two instances of this problem, the optimization on 20 block and 50 block stacks.

Whilst the 20 block instance falls well within the range of exhaustively testable problems, the latter already ranges amongst those which elude themselves from exhaustive testing. Therefore, prior to tackling the 50 block problem, a parameter optimization on the 20 block instance was conducted. During the following optimization runs, for both instances good stacks where found. For the 20 block problem the found stacks were close to optimal while those for the 50 block problem have been good but only comparable to the putative tight upper bound for 40 block stacks.

The problems during optimization of the latter instance mainly stem from the fact that too many instable stacks are generated both during initialization and

optimization—a problem that can be addressed in the future by coupling the proposed method with a local search procedure that is concerned with identifying stable stacks in the vicinity of a given, instable solution. Therefore, although the results not yet have been optimal, a proof of concept for the usefulness of the proposed method in tackling the given problem has been provided.

Acknowledgments

Tim Hohm has been supported by the European Commission under the Marie Curie Research Training Network SY-STEM, Project 5336.

References

1. J. G. Coffin. Problem 3009. *Amer Math Monthly*, 30(2):76, 1923.
2. O. Dubrovsky, G. Levitin, and M. Penn. A Genetic Algorithm with a Compact Solution Encoding for the Container Ship Stowage Problem. *Journal of Heuristics*, 8(6):585–599, 2002.
3. J. F. Hall. Fun with stacking blocks. *Am J Phys*, 73(12):1107–1116, 2005.
4. G. W. Housner and D. E. Hudson. *Applied Mechanics: Statics*. D. Van Nostrand, New York, NY, USA, 2 edition, 1961.
5. P. B. Johnson. Leaning Tower of Lire. *Am J Phys*, 23:240, 1955.
6. M. Paterson and U. Zwick. Overhang. In *SODA '06: Proceedings of the seventh annual ACM-SIAM symposium on Discrete Algorithms*, pages 231–240, New York, NY, USA, 2006. ACM Press.
7. H. C. K. Plummer. *The Principles of Mechanics: An Elementary Course*. G. Bell, London, UK, 1929.
8. W. Walton. *A collection of Problems in Illustration of the principles of Theoretical Mechanics*. Deighton, Bell, Cambridge, UK, 2 edition, 1855.