ANALYSIS OF UNCERTAINTY IN ENGINEERING DESIGN OPTIMIZATION PROBLEMS

Philipp Fleck[(a)], Michael Kommenda[(b)], Michael Affenzeller[(c)], Thorsten Prante[(d)]

[(a), (b), (c)] Heuristic and Evolutionary Algorithms Laboratory,
University of Applied Sciences Upper Austria, Softwarepark 11, 4232 Hagenberg i. M., Austria
[(b), (c)] Institute for Formal Verification,
Johannes Kepler University, Altenberger Straße 69, 4040 Linz, Austria
[(d)] V-Research GmbH,
CAMPUS V, Stadtstraße 33, 6850 Dornbirn

[(a)]philipp.fleck@fh-hagenberg.at, [(b)]michael.kommenda@fh-hagenberg.at,
[(c)]michael.affenzeller@fh-hagenberg.at, [(d)]thorsten.prante@v-research.at

ABSTRACT
In this paper, we analyze popular benchmark instances in the field of engineering design optimization regarding the robustness of published solutions. First, we implement selected benchmark problems with HeuristicLab and show the advantages of having a framework that enables rapid prototyping for optimization and analysis. Then, we show that many solutions quickly become infeasible when considering uncertainty like production inaccuracies. Based on these findings, we motivate why robust solutions for engineering design are important and present methods for measuring, identifying and visualizing robustness. Finally, we present how solutions can be compared and selected using a novel robustness measure.

Keywords: engineering design optimization, constraint handling, uncertainty, robustness

1. INTRODUCTION
The increasing complexity of today's engineering tasks have made computer systems an omnipresent part of the engineering design process. These tools help human experts to make meaningful design choices in order to develop products or systems that satisfy a complex set of requirements. Particularly, finding the optimal set of decisions for a given engineering design problem is an ongoing challenge.
Engineering design problems are often formulated as constrained optimization problems with a design vector $\boldsymbol{x} = (x_1, x_2, \dots, x_n) \in \mathbb{R}^n$, an objective function $f(\boldsymbol{x})$, inequality constraints $g_j(\boldsymbol{x}) \leq 0$ and equality constraints $h_k(\boldsymbol{x}) = 0$ with $f, g_j, h_k\colon \mathbb{R}^n \to \mathbb{R}$. The range of each design value x_i is either given explicitly, as $min_i \leq x_i \leq max_i$, or as inequality constraints; for example $1 \leq x_1 \leq 5$ is equivalent to $g_1 = 1 - x_1 \leq 0$ and $g_2 = x_1 - 5 \leq 0$.

A variety of benchmark engineering design optimization problems exists for evaluating and comparing different optimization techniques. In this paper, we cover four well studied benchmark problems, with formal descriptions of each problem given in the Appendix:

- Pressure Vessel (PV) (Sandgren 1988)
- Speed Reducer (SR) (Golinski 1973)
- Tension/Compression Spring (TCS) (Belegundu 1982)
- Welded Beam (WB) (Ragsdell and Phillips 1976)

We show how to implement the selected benchmark problems with the open source optimization framework HeuristicLab (Wagner et al. 2014) and demonstrate the advantages and simplicity of using a framework for rapid prototyping and compare our solutions to solutions from relevant literature. Then, we discuss uncertainty and present methods to quantify uncertainty and finally, we introduce analysis methods to compare the robustness of different solutions.

2. OPTIMIZATION OF ENGINEERING DESIGN PROBLEMS

2.1. Solving Engineering Design Optimization Problems with HeuristicLab
HeuristicLab (Wagner et al. 2014) is an open source framework for heuristic optimization that is freely available at http://dev.heuristiclab.com and offers a wide range of popular optimization algorithms and benchmark problems. Users can easily implement their own optimization problems, using the built-in scripting environment, which allows specifying the objective function of optimization problems in C# code directly in HeuristicLab (Scheibenpflug et al. 2015).

To implement a new problem, the user has to create a *Programmable Problem (single-objective)*, where some exemplary code is already provided. After having implemented the problem, the code is compiled at runtime and the problem can be solved using one of the many algorithms. A fully specified optimization problem requires three parts that the user has to implement (for illustration see Figure 1):

1. Specification whether the objective should be maximized or minimized, which is done by implementing the *Maximization* property.
2. An appropriate *Encoding* for the optimization problem. Code for popular encodings is already provided in the *Initialize* method, for instance the *RealVectorEncoding*.
3. The objective function, which is implemented in the *Evaluate* method and has to return the quality of a given solution.

Different forms of evaluating the objective are possible, ranging from implementing code for a simple analytical model to executing a full-scaled simulation. The benchmark optimization problems used in this paper specify analytical objective functions and are therefore quite easy to implement. For instance, the objective function for the tension/compression spring problem,

$$f(\boldsymbol{x}) = (x_3 + 2) \cdot x_2 \cdot x_1^2\,, \tag{1}$$

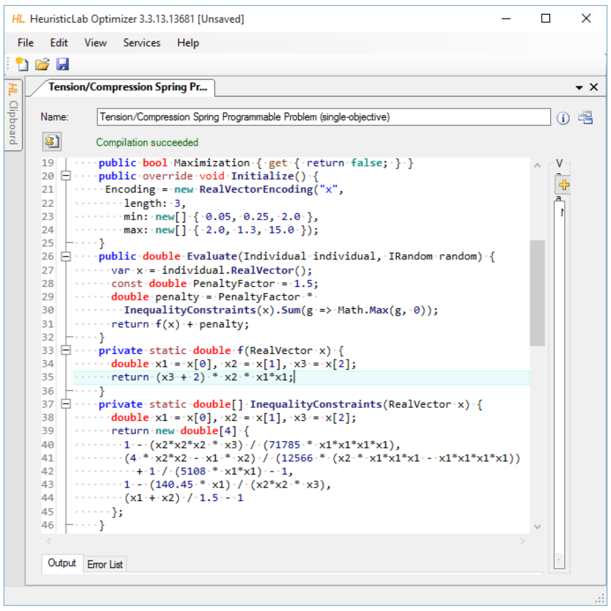
translates to the following source code (without constraint handling):

```
public double Evaluate(
    Individual ind, IRandom r) {
  var x = ind.RealVector();
  double x1=x[0], x2=x[1], x3=x[2];
  return (x3 + 2) * x2 * x1*x1;
}
```

Optionally, the *Analyze* method in the *Programmable Problem* can be implemented to gather additional information during the optimization, for instance recording the best feasible solution found. Figure 1 shows a screenshot of a fully specified optimization problem, including constraint handling.
Furthermore, HeuristicLab can be extended by developing new plugins, which allows the user to implement new encodings or new algorithms (see Scheibenpflug et al. (2015) for more details).

2.2. Constraint Handling

Many optimization problems specify different kinds of constraints. However, optimization algorithms usually do not handle constraints explicitly and require them being incorporated into the objective function. A common way of handling constraints are *penalty functions*, where the original constrained objective function is transformed into an unconstrained objective function by adding a penalty for infeasible solutions (Coello Coello 2002).


HeuristicLab Optimizer 3.3.13.13681 [Unsaved]
File
Edit
View
Services
Help
Clipboard
Tension/Compression Spring Pr...
Output
Error List

Name: Tension/Compression Spring Programmable Problem (single-objective)

Compilation succeeded

```
    public bool Maximization { get { return false; } }
    public override void Initialize() {
     Encoding = new RealVectorEncoding("x",
        length: 3,
        min: new[] { 0.05, 0.25, 2.0 },
        max: new[] { 2.0, 1.3, 15.0 });
    }
    public double Evaluate(Individual individual, IRandom random) {
      var x = individual.RealVector();
      const double PenaltyFactor = 1.5;
      double penalty = PenaltyFactor *
        InequalityConstraints(x).Sum(g => Math.Max(g, 0));
      return f(x) + penalty;
    }
    private static double f(RealVector x) {
      double x1 = x[0], x2 = x[1], x3 = x[2];
      return (x3 + 2) * x2 * x1*x1;
    }
    private static double[] InequalityConstraints(RealVector x) {
      double x1 = x[0], x2 = x[1], x3 = x[2];
      return new double[4] {
        1 - (x2*x2*x2 * x3) / (71785 * x1*x1*x1*x1),
        (4 * x2*x2 - x1 * x2) / (12566 * (x2 * x1*x1*x1 - x1*x1*x1*x1))
          + 1 / (5108 * x1*x1) - 1,
        1 - (140.45 * x1) / (x2*x2 * x3),
        (x1 + x2) / 1.5 - 1
      };
    }
```

Figure 1: Screenshot showing the implementation of the tension/compression spring problem with HeuristicLab's integrated scripting environment.

A new unconstrained objective function $\phi(\boldsymbol{x})$ is often formulated as

$$\phi(\boldsymbol{x}) = f(\boldsymbol{x}) \pm \left(\sum\nolimits_{j=1} r_j G_j + \sum\nolimits_{k=1} c_k H_k\right) \tag{2}$$

where r_j and c_k are constants called *penalty factors*. G_j and H_k are functions of the inequality and equality constraints $g_j(\boldsymbol{x})$ and $h_k(\boldsymbol{x})$ respectively and are commonly defined as

$$G_j = \max\big(0, g_j(\boldsymbol{x})\big)^{\beta} \quad \text{and} \quad H_k = |h_k(\boldsymbol{x})|^{\gamma} \tag{3}$$

where β and γ are usually 1 or 2. The penalties are added if the problem is minimized, or subtracted if maximized. All problems solved in this paper are minimization problems with only inequality constraints; further, we simplify problem specification by only a single penalty value r. The resulting objective function used further in this paper is

$$\phi(\boldsymbol{x}) = f(\boldsymbol{x}) + r \sum\nolimits_{j=1} \max\big(0, g_j(\boldsymbol{x})\big)\ . \tag{4}$$

For each optimization problem, the penalty factor r is determined by obtaining a typical objective value for this problem and multiplying it by a factor to make it intentionally worse. We obtained the typical objective value by calculating the mean of 10,000 random solutions and then multiplying it by 5. For each problem, the resulting penalty value is stated in Table 1.

2.3. Single-Objective Optimization

We use the Covariance Matrix Adaption Evolution Strategy (CMA-ES) to solve the engineering design optimization problems. The CMA-ES is a population-based evolutionary algorithm, specifically designed for real-valued, single objective optimization problems (Hansen et al. 2003).

The basic idea of the algorithm is to sample promising areas in the search space with a multivariate normal distribution $\mathcal{N}(\boldsymbol{m}, \sigma^2 C)$, defined by the mean vector $\boldsymbol{m} \in \mathbb{R}^n$ and the covariance matrix $\sigma^2 C \in \mathbb{R}^{n \times n}$ where $\sigma \in \mathbb{R}$ defines an additional scaling factor (called the step-size). Figure 2 illustrates different types of distributions by different covariance matrices.

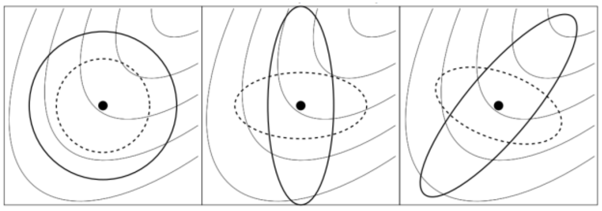


Figure 2: Different distributions given by the same mean vector but different covariance matrices, symbolized by density-ellipsoids: The identity covariance matrix resulting in circular distributions (left). A diagonal covariance matrix resulting in axis-aligned distributions (middle). A symmetric, positive-definite matrix capable of describing axis-independent distributions (right). The background shows the gradients of the search space towards the optimum in the upper right. Figure adapted from Hansen (2005).

The algorithm maintains and updates the parameters of the multivariate normal distribution so that it increases the probability of moving towards promising areas in the search space. Creating new sample points and updating the parameters is done iteratively, until a termination criteria is met. The following briefly describes the main parts of the CMA-ES. An extensive description is given in Hansen et al. (2003).

(1) In the beginning, the first mean vector $\boldsymbol{m}$ is determined by randomly creating λ vectors and calculating their mean. The initial step-size σ is given by the user, usually as a vector to setup axis-scaling for each dimension. The matrix C is initialized with the identity matrix.
(2) At each iteration, λ samples are created with the multivariate normal distribution $\mathcal{N}(\boldsymbol{m}, \sigma^2 C)$, with the mean vector $\boldsymbol{m}$ describing the current center of search and the covariance matrix $\sigma^2 C$ describing the form of the distribution and thus the search direction. The step-size σ controls the scale, therefore the "speed" the algorithm moves through the search space.
(3) The solutions are evaluated according to the given objective function and sorted based on their objective value.
(4) The weighted sum of the μ best solutions is calculated to determine the new mean vector $\boldsymbol{m}$. The weights, which must sum to one, are either equal or decrease to favor better solutions.
(5) The step-size σ is updated based on the 1/5th success rule already used in a standard evolution strategy (Schumer and Steiglitz 1968). The matrix C is updated based on the gradient of the quality, approximated by the sampled solutions.

2.4. Results

We implemented and optimized the four selected engineering design optimization problems with the techniques described in the previous sections. For all problems we used the CMA-ES with a population size (λ) of 50 and 25 individuals (μ) for calculating the new mean (log-scaled weighting). After a maximum of a 100 iterations, the algorithm terminates. Table 1 shows the initial step-size σ and penalty factor r for each problem.

Table 1: Parameters used for the different problems.

Probl.	Initial σ	r
PV	(2, 2, 63, 63)	65000
SR	(0.33, 0.033, 3.67, 0.33, 0.33, 0.33, 0.167)	20000
TCS	(0.65, 0.35, 4.33)	50
WB	(0.63, 3.3, 3.3, 0.63)	70

Table 2 contains a comparison between our results and the results of various literature, showing that the results of the CMA-ES are equal or close to the best-found solutions from the literature. This demonstrates that common benchmark optimization problems in the fields of engineering design can be solved with little implementation effort, when using an appropriate framework that allows easy problem specification and offers proper optimization algorithms. Otherwise, implementing the problem specification and optimization algorithms can be time consuming.

While recalculating the quality of the solutions from the literature for validation, we observed that several solutions slightly violate some of the constraints. We suspect that the published solutions are rounded to some degree, causing the small infeasibilities in our calculations. Hence, we introduced a tolerance value for evaluating inequality constraints, so that $g_j(\boldsymbol{x}) \leq 10^{\epsilon}$, to determine how closely the constraints are violated. Table 2 includes the number of violated constraints per solution and states the tolerance value so that the solution becomes feasible. Not all authors reported solutions for all four problems we analyze, thus some entries in the table are empty.

For readability, the numbers in Table 2 are rounded, yielding solutions with the same quality but different constraint violations. A detailed table with the precise qualities and the all solution vectors is available online at http://dev.heuristiclab.com/AdditionalMaterial#EMSS2016.

Additionally, all implemented problems and fully configured optimization algorithms along with the algorithm results presented in this paper are also available under the same URL.

For some problems multiple versions exist, e.g. Welded Beam, thus we only compare results that were solved using the exact same problem specification. The exact specifications used in this paper can be found in the Appendix.

Table 2: Results of selected engineering design problems solved by the CMA-ES (present study) compared with other selected literature solutions. "Quality" shows the quality (rounded) of the best solution found, "V" shows the number of violated constraints for that solution and "ϵ" the tolerance factor (in 10^{ϵ}) to make the solution feasible. The best and best feasible solution are marked bold.

Reference	Pressure Vessel			Speed Reducer			Tension/C. Spring			Welded Beam		
	Quality	V	ϵ	Quality	V	ϵ	Quality	V	ϵ	Quality	V	ϵ
Present study	**6059.71**	**0**		2994.47	0		0.0126652	0		**2.38096**	**0**	
(Akay and Karaboga 2010)	6059.71	1	-8	2997.06	1	-6	0.0126650	1	-4			
(Akhtar et al. 2002)	6171.00	0		3008.08	0					2.44260	0	
(Azad and Fernandes 2011)	6059.53	2	0	**2994.32**	**2**	**-3**	0.0126640	2	-4	**2.38081**	**1**	**-4**
(Bernardino et al. 2007)	6060.37	0		2994.47	1	-6	0.0126680	0		2.38122	0	
(Bernardino et al. 2007)	6060.14	0		2994.47	0		0.0126660	0		2.38125	0	
(Bernardino et al. 2008)	6059.85	0		2996.35	1	-6	0.0126660	1	-5	2.38144	0	
(Bernardino et al. 2008)	6065.82	0		2996.35	1	-6	0.0126838	0		2.38335	0	
(Bernardino et al. 2008)	6832.58	1	1	2996.35	1	-6	0.0126790	0		2.59610	0	
(Cagnina et al. 2008)	6059.71	1	-1	2996.35	2	-6	0.0126650	1	-4			
(Coello Coello 2000)	6288.74	0					0.0127048	0				
(He and Wang 2007)	6061.08	0					0.0126747	0				
(He et al. 2004)	6059.71	1	-10				0.0126653	1	-7	**2.38096**	**0**	
(Hu et al. 2003)	6059.13	1	-7				0.0126661	1	-8			
(Kaveh and Talatahari 2009)	**6059.09**	**1**	**-4**				**0.0126391**	**1**	**-2**			
(Kaveh and Talatahari 2010)	6059.73	0					0.0126432	1	-2			
(Lemonge et al. 2010)	6059.72	1	1	2996.35	1	-6	0.0126790	1	-5	2.38124	0	
(Mezura Montes et al. 2003)	6059.71	1	1	3025.01	0		**0.0126650**	**0**				
(Mezura Montes et al. 2007)	6059.70	2	1	2996.36	0		0.0126980	0				
(Ray and Liew 2003)				2994.74	0		0.0126692	0		2.38543	0	
(Rocha and Fernandes 2009)	6071.17	0		**2994.37**	**0**		0.0126680	0		2.38627	0	
(Rocha and Fernandes 2009)	6072.23	0		2995.80	0		0.0126670	0		2.43162	0	
(Tomassetti 2010)	6059.71	1	-8	2996.35	1	-6	**0.0126650**	**0**				
(Zhang et al. 2008)	7197.70	1	1	2994.47	0		0.0126652	1	-8	2.38096	0	

3. ROBUSTNESS IN ENGINEERING DESIGN

Our results in Section 2.4 show that small deviations can make solutions infeasible. In this section, we generalize these small deviations by applying *uncertainty* in various stages of the engineering design process to observe how the final quality and feasibility of solutions changes. Our goal is to find *robust* solutions that stay feasible for increasing uncertainty while maintaining high quality.

To analyze robustness, we first define different types of uncertainty and describe how to quantify them. Then, we introduce methods for measuring and visualizing robustness that allows comparing different solutions. Finally, we introduce a novel measure that allows ranking of solutions based on their robustness.

3.1. Uncertainty characterization

Uncertainty summarizes indirect influences that can occur during the engineering design process and are not controllable by the design vector $\boldsymbol{x}$. Beyer and Sendhoff (2007) describe four types of uncertainty, based on which aspect of evaluation is influenced.

A) Changing environmental and operating conditions summarize controllable influences of the evaluation process that are not reflected in the design variables. Typical influences are environmental settings like operating temperature.

The objective function f becomes a function of the design vector $\boldsymbol{x}$ and an additional parameter vector $\boldsymbol{\alpha}$ that quantifies all additional settings of the system,

$$f = f(\boldsymbol{x}, \boldsymbol{\alpha})\,. \tag{5}$$

B) Production tolerances and actuator imprecision summarize uncontrollable influences on the design vector. For instance, machine imprecision when actually manufacturing a product described by the design vector. Formally, a perturbation vector $\boldsymbol{\delta}$ is added on the design vector, resulting in an objective function

$$f = f(\boldsymbol{x} + \boldsymbol{\delta}, \boldsymbol{\alpha})\,, \tag{6}$$

where the perturbation is absolute or relative ($\boldsymbol{\delta} = \boldsymbol{\epsilon} * \boldsymbol{x}$).

C) Uncertainty in the system output occurs due to imprecise measuring of the output. The observed output differs from the actual evaluation output, which is formally described by a random function

$$\tilde{f} = \tilde{f}[f(\boldsymbol{x} + \boldsymbol{\delta}, \boldsymbol{\alpha})]\,. \tag{7}$$

D) Feasibility uncertainties summarize effects on constraint satisfaction. Technically, this is not a separate type of uncertainty and is often modeled as type A or type B.

In the remainder of this paper, we only use uncertainty type B, *production tolerances*, because they can be specified in the same way for all engineering design problems and they are best suited to demonstrate uncertainty in this paper.

3.2. Quantifying Uncertainty

Uncertainties can be quantified *deterministically*, *probabilistically* or *possibilistically* (Beyer and Sendhoff 2007). We quantify uncertainties probabilistically, using normal distributions $\mathcal{N}(\mu, \sigma^2)$ and uniform distributions $\mathcal{U}(a, b)$ to describe the perturbation vector $\boldsymbol{\delta}$, which we define relatively to the design vector $\boldsymbol{x}$. For instance, $\boldsymbol{\delta}_i = 0.01\,\boldsymbol{x}_i$ increases an entry of the design vector by one percent. Quantifying the uncertainty then becomes a matter of quantifying the underlying random distributions.

Usually, uncertainties affect production processes on different levels and with different magnitudes. Therefore, each entry of the perturbation vector can be separately defined by its own probability distribution, or the whole perturbation vector is defined by a multivariate random distribution. However, for the remainder of this paper, we simply define all entries with the same distribution, except otherwise specified.

Because the perturbation is defined relative, and we do not want to introduce an intentional bias, the random distributions are parameterized to have their center at zero. This reduces the parameters of the distribution to a single value: the variance σ^2 of the normal distribution $\mathcal{N}(0, \sigma^2)$ or the range r of the uniform distribution $\mathcal{U}(-r/2, +r/2)$. We call this single parameter *uncertainty level* u. For instance, using a uniformly distributed uncertainty with an uncertainty level $u = 10^{-2}$ describes the uncertainty using $\mathcal{U}(-0.5, +0.5)$.

3.3. Measuring Robustness

Evaluating a solution at a fixed uncertainty level does not give a general measure of how a solution behaves under a broader range of uncertainty. Instead, an interval of uncertainty levels must be evaluated, with the lower and upper bounds depending on the robustness requirements of the optimization problem. If the production process is not known, we use larger uncertainty ranges, e.g. between 10^{-8} and 10^{-1}. If the expected uncertainty is known, the range should be set more precisely.

For each uncertainty level within the range, the objective value (original objective without penalties) and the constraint violations are calculated, yielding progressions of the objective values and constraint violations over increasing uncertainties. In the first step, we analyze robustness only with respect to the progression of constraint violations.

Because $\boldsymbol{\delta}$ is a random variable, constraint violations are not binary anymore, but measured as a probability instead; therefore, for each constraint, a progression of the violation probability is obtained with values between zero and one. We aggregate these violation progressions by calculating the sum of all violation probabilities at each uncertainty level, resulting in a single violations progression with each value between zero (no violations) and the number of constraints (all constraints violated). Although aggregating prevents analyzing which specific constraints were violated, this is neglected for a general measure of robustness.

Figure 3 shows an example of four different solutions for the welded beam problem, with uncertainties ranging from 10^{-8} to 1. The uncertainties on the x-axis are scaled logarithmically to be able to observe effects on lower uncertainty levels as well as on higher levels. Because we name solutions by their literature origin, we use italic font when referring to solutions to avoid confusion with regular literature cites.

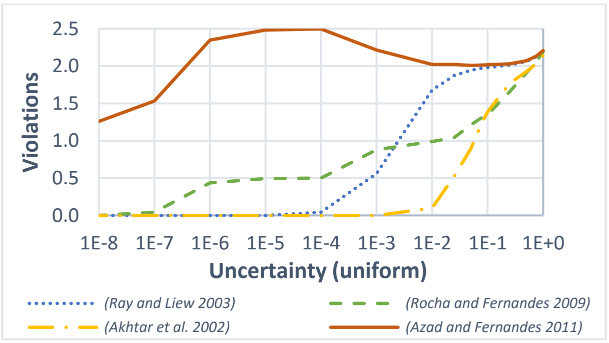


Figure 3: Progressions of violations over increasing uncertainties for selected solutions of the welded beam problem.

Typically, the violations begin to increase at different uncertainty levels for different solutions, e.g. *(Rocha and Fernandes 2009)* around 10^{-7} and *(Ray and Liew 2003)* around 10^{-4}. Robust solutions show influences only at higher uncertainty, whereas fragile solutions are affected earlier. The violations need not be strictly increasing, c.f. *(Azad and Fernandes 2011)*. This can be the case if the solution violates constraints at lower or no uncertainty, and sufficiently high uncertainty allows it to become feasible again.

The series of violations also allows visual comparisons of different solutions. First and most important, solutions with a progression strictly lower than the progression of an other solution are more robust since they violate less constraints at all uncertainty levels, cf. *(Akhtar et al. 2002)* versus *(Ray and Liew 2003)*. This allows ranking solutions based on their robustness. In addition, the violations-gap of two solutions at a specific uncertainty level tells how much more robust a solution is.

Solutions with intersecting series cannot be compared unambiguously, like *(Rocha and Fernandes 2009)* and *(Ray and Liew 2003)*. In such a case, unambiguity is limited to non-intersecting intervals. The smaller the interval, the more likely that multiple solutions can be compared unambiguously.

3.4. Estimate Robustness Ranking

We still want to estimate which solution is more robust over a large uncertainty interval, even though comparing solutions unambiguously is not possible when their violations progressions intersect. We calculate the area V

under the curve of the violations series over the uncertainty range. Because we do not know whether higher or lower uncertainty is more important, we weight them equally by using a log-scale on the uncertainty-axis, as in Figure 3. We use the middle Riemann sum with a log-scaled partition size to calculate the violation area

$$V = \sum_{i=1}^{n} \frac{v(\boldsymbol{x}, u_i) + v(\boldsymbol{x}, u_{i-1})}{2} (\log u_i - \log u_{i-1}) , \quad (8)$$

where $v(\boldsymbol{x}, u)$ yields the number of violated constraints for a solution $\boldsymbol{x}$ at the uncertainty level u and n is the number of samples drawn from the uncertainty range.
We argue that solutions with a lower area of violations are more robust over a larger uncertainty interval. This gives a single, real-valued coefficient for robustness that allows ordering and distance calculation.
However, this coefficient cannot be interpreted easily because its magnitude depends on the uncertainty range. By dividing the area by the range of the (logarithmized) uncertainty interval, it can be directly related to values calculated by a fixed uncertainty level. This relative area of violations,

$$V_{rel} = \frac{V}{\log u_{max} - \log u_{min}} , \quad (9)$$

can also be interpreted as weighted mean of violations at multiple fixed uncertainty levels, where the weights are the distances of two adjacent uncertainty levels.
With a single coefficient that describes robustness, we are able to compare quality and robustness in a scatterplot. This allows visualizing the tradeoff between robustness and quality, as shown in Figure 4 with selected solutions for the speed reducer problem. The objective is calculated similarly by the area under the objective progression. As the objective and the violations are minimized, solutions at the lower left are preferable. However, most solutions are either robust and low quality, like *(Mezura Montes et al. 2003)*, or high quality and not robust, like *(Bernardino et al. 2007)*. *(Rocha and Fernandes 2009)* would be a reasonable tradeoff.

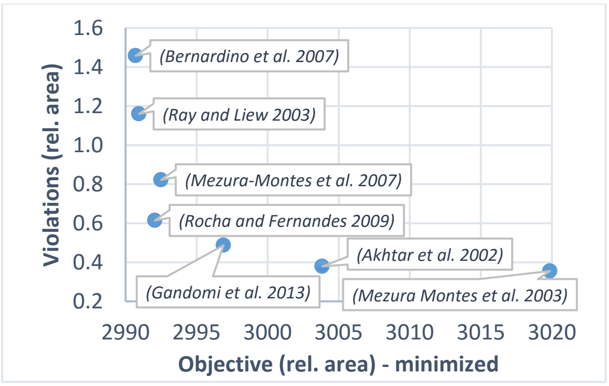


Figure 4: Comparison of objective and robustness of different solutions of the speed reducer problem by a scatterplot.

3.5. Results

In this section we present selected results from different problems for an overview. The full analysis of all problems with all solutions can be downloaded at http://dev.heuristiclab.com/AdditionalMaterial#EMSS2016.
For all the presented results we used uniformly distributed uncertainties, because it turned out the results are similar to using normal distributions and uniform distributions are easier to interpret due to the fixed ranges.
Concerning quality, we observed two general behaviors, demonstrated by Figure 5. First, uncertainty significantly influences the quality only at higher uncertainty levels with $u > 10^{-1}$; thus, at lower uncertainty, the order of solutions by their quality does not change at all. Secondly, the tendency whether quality increases or decreases with increasing uncertainty is the same for all solutions of a problem. Even the extend of the quality changes is very similar. This concludes that uncertainty does not significantly influence the order of solutions by their quality.

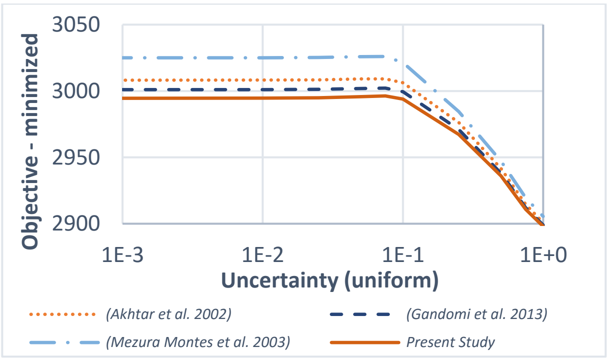


Figure 5: Quality progression over uncertainty for solutions of the speed reducer problem.

More interesting are the constraints violations, where we start with the results of the tension compression spring problem, since the results are easier to interpret. Figure 6 shows the violations progressions of selected solutions between uncertainty level 10^{-10} and 10^{-1}. Only solutions that did not violate any constraints without uncertainty were selected (see Table 2). Because there are effectively no intersections between the progressions, an unambiguous ranking for robustness can be determined. Our own solution, found by a CMA-ES, starts violating constraints early ($u < 10^{-10}$) because the solution lies very close to constraint boundaries. This also demonstrates that solutions specifically optimized for maximum quality are typically not robust. The other solutions from the literature use less significant decimal places thus are generally not as close to constraint boundaries. The most robust solution from the selected solutions in Figure 6 is *(Coello Coello 2000)*; however, when compared by quality, this solution performs worse than all others, shown in the scatterplot in Figure 8. For quality only, *(Ray and Liew 2003)* would be the better choice. Choosing an appropriate compromise depends mainly on the expected uncertainty and specifics of the optimization problem.

The progressions of violations for the pressure vessel problem, shown in Figure 7, is more difficult to interpret. First of all, intersecting progressions means that not all solutions can be compared unambiguously, for instance *(Akhtar et al. 2002)* and *(Coello Coello 2000)*. Some solutions, however, can be ranked unambiguously, for instance *(He and Wang 2007)* is more robust on all levels than *(Bernardino et al. 2007)*, *(Kaveh and Talatahari 2010)* and the solution from this paper by the CMA-ES. Similar to the results of the tension compression spring, our solution is less robust than all other solutions.
Choosing the most robust solution in this case depends on the expected uncertainty. If one expects small uncertainties $u < 10^{-2}$, the most robust solution would be *(Akhtar et al. 2002)*. If the production system usually inflicts high uncertainty ($u > 2 \cdot 10^{-2}$), *(Coello Coello 2000)* would be better. The estimated ranking over the whole uncertainty range, as described earlier, can be deduced from the scatterplot in Figure 9. The scatterplot also shows that *(Akhtar et al. 2002)* is slightly more robust than *(Coello Coello 2000)*, but both have a significantly worse quality than all other displayed solutions.
Results for the speed reducer problem and welded beam problem are omitted in this paper because the results are similar to the pressure vessel problem.

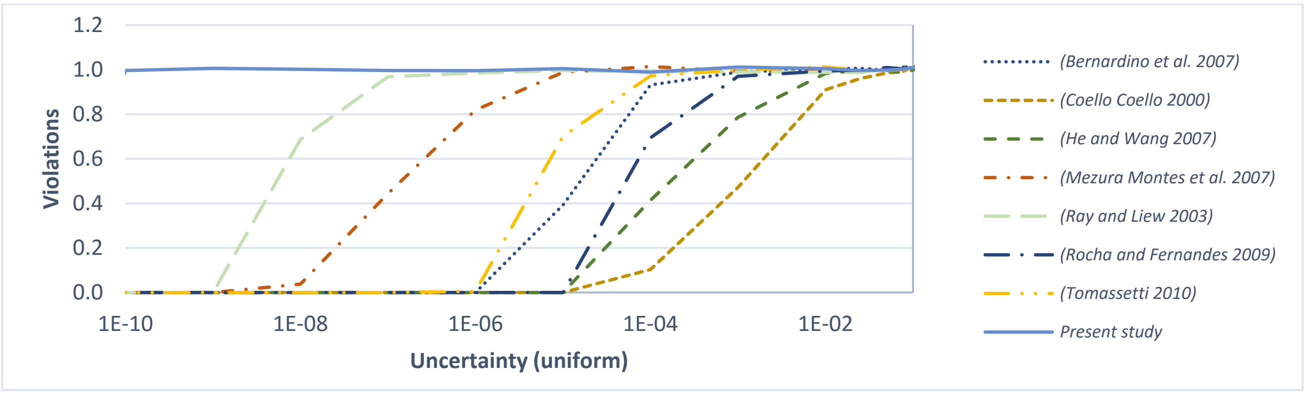


Figure 6: Violations progression over uncertainty for selected solutions of the tension compression spring problem.

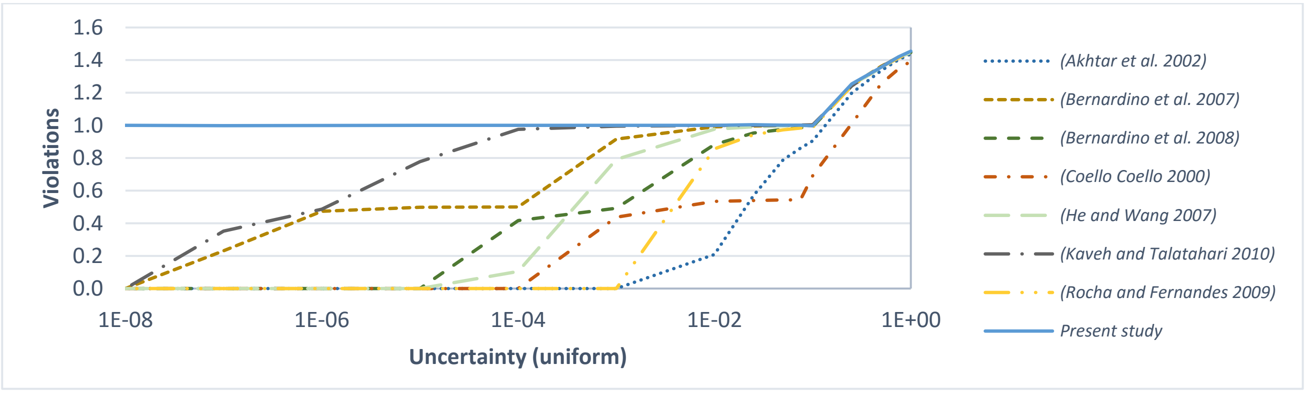


Figure 7: Violations progression over uncertainty for selected solutions of the pressure vessel problem.

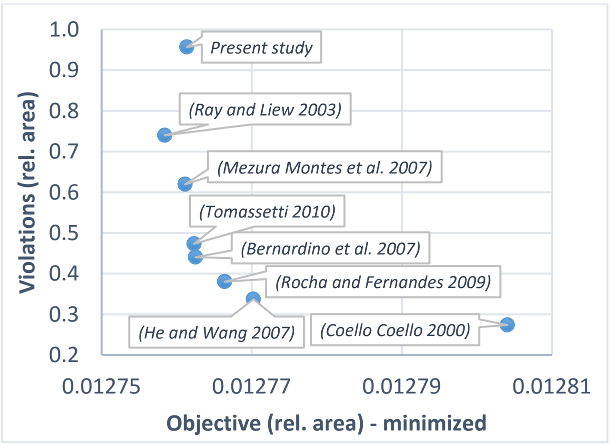


Figure 8: Scatterplot showing quality vs. violations of selected solutions for the tension compression spring problem.

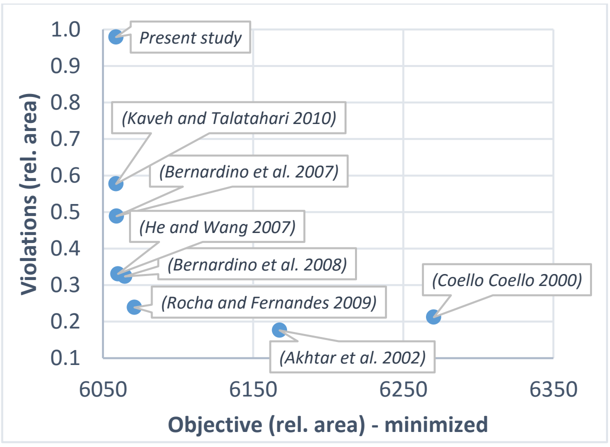


Figure 9: Scatterplot showing quality vs. violations of selected solutions for the pressure vessel problem.

4. DISCUSSION

In this paper, we motivated robustness of solutions in engineering design problems and demonstrated the necessary steps to analyze uncertainty and robustness.

In the first part of this paper, we implemented popular constrained optimization problems using HeuristicLab and demonstrated different techniques for constraint handling by transforming the problem into an unconstrained optimization problem using penalties. Then, we used the CMA-ES to find solutions for the given problems and compared our solutions to solutions from various literature sources.

In the second part of this paper, we introduced uncertainty as an influence that causes small perturbations to solution candidates. We quantified these uncertainties with random distributions and introduced uncertainty levels as a measure of the variance uncertainty inflicts on the solutions. Then, we showed that the robustness of solutions can be visualized and compared using charts that plot the progression of violated constraints over increasing uncertainty and further demonstrated how the area under the progression can be used to estimate a ranking of robustness when an unambiguous ranking is not possible. Finally, we illustrated that a solution's quality and robustness often oppose and a compromise must be picked.

A reasonable next step would be to consider uncertainty directly during the solving and optimization process, to search for solutions that are high quality and are robust.

This suggests using multi-objective optimization techniques, which simultaneously optimize the objective value of a solution as well as its robustness. However, new questions arise, for instance, for which uncertainty level a solution should be optimized for? Is it possible to optimize it for a wider range of uncertainty? Which (optimization) algorithms are to use for optimizing the resulting multi-objective optimization problem?

Another interesting aspect for future research would be the analytical prediction of robustness based on the formal description of a constrained optimization problem. Our research suggested that for the same optimization problem, increasing uncertainty has the same effect on the quality for all solutions. Those correlations could also be predicted analytically by analyzing the objective function in detail. Similarly, the robustness of a solution could be predicted analytically by analyzing the constraints, how close a solution is to those constraints and how perturbation would increase the likelihood that those constraints will be violated.

For a broader empirical confirmation of our findings, additional engineering design optimization problems could be studied. A real world test scenario would also be very interesting.

We believe that uncertainty, and the goal to find robust solutions, will become a major factor in engineering problems. Thus, the study of techniques and algorithms that are able to find robust solutions while maintaining high quality will continue and increase.

ACKNOWLEDGMENTS

The work described in this paper was done within the COMET Project #843551 Advanced Engineering Design Automation (AEDA) and COMET Project #843532 Heuristic Optimization in Production and Logistics (HOPL), both funded by the Austrian Research Promotion Agency (FFG).

APPENDIX A: BENCHMARK INSTANCES

A.1 Pressure Vessel Problem

Minimize the fabrication costs of a cylindrical pressure vessel given by the thickness of the pressure vessel (x_1), the thickness of the head (x_2), the inner radius of the vessel (x_3) and the length of the vessel without heads (x_4), shown in Figure 10.

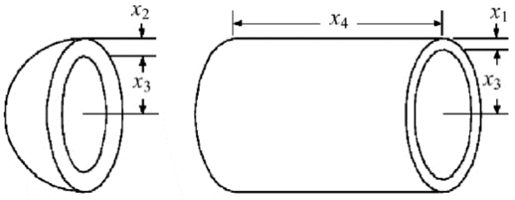


Figure 10: Schematic illustration of a pressure vessel and its parameters. Image adapted from Cagnina et al. (2008).

The pressure vessel problem is formally described as

$$f(\boldsymbol{x}) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

where

$$\boldsymbol{x} = (x_1, x_2, x_3, x_4) \in \mathbb{R}^4$$
$$x_1 = \{0.0625t \mid t \in \mathbb{Z} \cap [1;99]\}$$
$$x_2 = \{0.0625t \mid t \in \mathbb{Z} \cap [1;99]\}$$
$$x_3 \in [10;200] \quad x_4 \in [10;200]$$

subject to $g_j(\boldsymbol{x}) \leq 0$ with

$$g_1(\boldsymbol{x}) = -x_1 + 0.0193x_3 \quad g_2(\boldsymbol{x}) = -x_2 + 0.00954x_3$$
$$g_3(\boldsymbol{x}) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1{,}296{,}000 \quad g_4(\boldsymbol{x}) = x_4 - 240$$

A.2 Speed Reducer Problem

Minimize the weight of a speed reducer given by the gear face width (x_1), the teeth module (x_2), the number of pinion teeth (x_3), the lengths of the shafts between bearings (x_4, x_5) and the diameters of the shafts (x_6, x_7), shown in Figure 11. The speed reducer is constrained to bending stress of the gear teeth, surface stress, transverse deflection of the shafts and stress in the shafts.

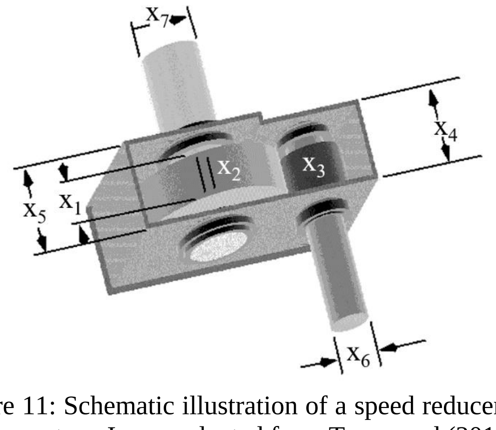


Figure 11: Schematic illustration of a speed reducer and its parameters. Image adapted from Townsend (2016).

The speed reducer problem is formally described as

$$f(\boldsymbol{x}) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2)$$

where

$$\boldsymbol{x} = (x_1, x_2, \dots, x_7) \in \mathbb{R}^7$$
$$x_1 \in [2.6; 3.6] \quad x_2 \in [0.7; 0.8] \quad x_3 \in \mathbb{Z} \cap [17; 28]$$
$$x_4 \in [7.3; 8.3] \quad x_5 \in [7.3; 8.3] \quad x_6 \in [2.9; 3.9]$$
$$x_7 \in [5; 5.5]$$

subject to $g_j(\boldsymbol{x}) \le 0$ with

$$g_1(\boldsymbol{x}) = \frac{27}{x_1x_2^2x_3} - 1 \quad g_2(\boldsymbol{x}) = \frac{397.5}{x_1x_2^2x_3^2} - 1$$
$$g_3(\boldsymbol{x}) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \quad g_4(\boldsymbol{x}) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1$$
$$g_5(\boldsymbol{x}) = \frac{1}{110x_6^3}\sqrt{\left(\frac{745x_4}{x_2x_3}\right)^2 + 16.9 \cdot 10^6} - 1$$
$$g_6(\boldsymbol{x}) = \frac{1}{85x_7^3}\sqrt{\left(\frac{745x_5}{x_2x_3}\right)^2 + 157.5 \cdot 10^6} - 1$$
$$g_7(\boldsymbol{x}) = \frac{x_2x_3}{40} - 1 \quad g_8(\boldsymbol{x}) = \frac{5x_2}{x_1} - 1$$
$$g_9(\boldsymbol{x}) = \frac{x_1}{12x_2} - 1 \quad g_{10}(\boldsymbol{x}) = \frac{1.5x_6 + 1.9}{x_4} - 1$$
$$g_{11}(\boldsymbol{x}) = \frac{1.1x_7 + 1.9}{x_5} - 1$$

A.3 Tension/Compression Spring Problem

Minimize the weight of a coil spring under a constant tension/compression load given by the wire diameter (x_1), the winding diameter (x_2) and the number of active coils (x_3), shown in Figure 12.

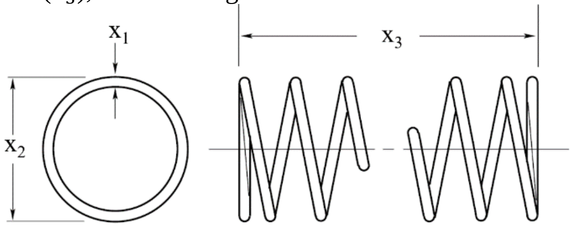


Figure 12: Schematic illustration of a tension/compression spring and its parameters. Image adapted from Bernardino et al. (2007).

The tension/compression spring problem is formally described as

$$f(\boldsymbol{x}) = (x_3 + 2)x_2x_1^2$$

where

$$\boldsymbol{x} = (x_1, x_2, x_3) \in \mathbb{R}^3$$
$$x_1 \in [0.05; 2] \quad x_2 \in [0.25; 1.3] \quad x_3 \in [2; 15]$$

subject to $g_j(\boldsymbol{x}) \le 0$ with

$$g_1(\boldsymbol{x}) = 1 - \frac{x_2^3x_3}{71{,}785x_1^4}$$
$$g_2(\boldsymbol{x}) = \frac{4x_2^2 - x_1x_2}{12{,}566(x_2x_1^3 - x_1^4)} + \frac{1}{5{,}108x_1^2} - 1$$
$$g_3(\boldsymbol{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \quad g_4(\boldsymbol{x}) = \frac{x_1 + x_2}{1.5} - 1$$

A.4 Welded Beam Problem

Minimize the fabrication costs of a welded beam given by the thickness of the weld (x_1), the length of the welded joint (x_2), the width of the beam (x_3) and the thickness of the beam (x_4), shown in Figure 13. The welded beam is subject to constraints on shear stress (τ), bending stress in the beam (σ), buckling load on the bar (P_c) and end deflection of the beam (δ).

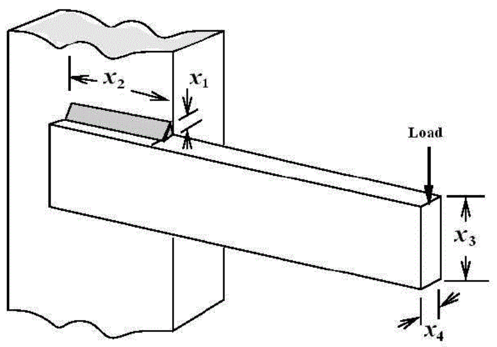


Figure 13: Schematic illustration of a welded beam and its parameters. Image adapted from Cagnina et al. (2008).

The welded beam problem is formally described as

$$f(\boldsymbol{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2)$$

where

$$\boldsymbol{x} = (x_1, x_2, x_3, x_4) \in \mathbb{R}^4$$
$$x_1 \in [0.1; 2] \quad x_2 \in [0.1; 10] \quad x_3 \in [0.1; 10] \quad x_4 \in [0.1; 2]$$

subject to $g_j(\boldsymbol{x}) \le 0$ with

$$g_1(\boldsymbol{x}) = \tau(x) - \tau_{\max} \quad g_2(\boldsymbol{x}) = \sigma(x) - \sigma_{\max}$$
$$g_3(\boldsymbol{x}) = x_1 - x_4 \quad g_4(\boldsymbol{x}) = 0.125 - x_1$$
$$g_5(\boldsymbol{x}) = \delta(x) - \delta_{max} \quad g_6(\boldsymbol{x}) = P - P_c(x)$$

and

$$\tau(\boldsymbol{x}) = \sqrt{\tau_1^2 + 2\tau_1\tau_2\frac{x_2}{2R} + \tau_2^2} \quad \tau_1 = \frac{P}{\sqrt{2}x_1x_2} \quad \tau_2 = \frac{MR}{J}$$
$$M = P\left(L + \frac{x_2}{2}\right) \quad R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}$$
$$J = 2\frac{x_1x_2}{\sqrt{2}}\left(\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right)$$
$$\sigma(\boldsymbol{x}) = \frac{6PL}{x_4x_3^2} \quad \delta(\boldsymbol{x}) = \frac{4PL^3}{Ex_3^3x_4}$$
$$P_c(\boldsymbol{x}) = \frac{4.013\sqrt{\frac{EGx_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$$
$$\tau_{\max} = 13{,}600 \text{ psi} \quad \sigma_{\max} = 30{,}000 \text{ psi} \quad \delta_{max} = 0.25 \text{ in}$$
$$P = 6{,}000 \text{ lb} \quad G = 12 \cdot 10^6 \text{ psi} \quad E = 30 \cdot 10^6 \text{ psi}$$
$$L = 14 \text{ in}$$

REFERENCES

Akay B, Karaboga D (2010) Artificial bee colony algorithm for large-scale problems and engineering design optimization. Journal of Intelligent Manufacturing 23(4):1001–1014

Akhtar S, Tai K, Ray T (2002) A socio-behavioural simulation model for engineering design optimization. Engineering Optimization 34(4):341–354

Azad MAK, Fernandes EM (2011) Modified Differential Evolution Based on Global Competitive Ranking for Engineering Design Optimization Problems. In: Computational Science and its Applications, vol 6784, pp 245–260

Belegundu AD (1982) Study of mathematical programming methods for structural optimization. PhD thesis

Bernardino HS, Barbosa HJ, Lemonge AC (2007) A hybrid genetic algorithm for constrained optimization problems in mechanical engineering. In: IEEE Congress on Evolutionary Computation, pp 646–653

Bernardino HS, Barbosa HJ, Lemonge AC, Fonseca LG (2008) A new hybrid AIS-GA for constrained optimization problems in mechanical engineering. In: IEEE Congress on Evolutionary Computation, pp 1455–1462

Beyer H, Sendhoff B (2007) Robust optimization – A comprehensive survey. Computer Methods in Applied Mechanics and Engineering 196(33-34):3190–3218

Cagnina LC, Esquivel SC, Coello Coello CA (2008) Solving engineering optimization problems with the simple constrained particle swarm optimizer. Informatica 32(3):319–326

Coello Coello CA (2000) Use of a self-adaptive penalty approach for engineering optimization problems. Computers in Industry 41(2):113–127

Coello Coello CA (2002) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. Computer Methods in Applied Mechanics and Engineering 191(11-12):1245–1287

Golinski J (1973) An adaptive optimization system applied to machine synthesis. Mechanism and Machine Theory 8(4):419–436

Hansen N (2005) The CMA evolution strategy: A tutorial. Vu le 29

Hansen N, Müller SD, Koumoutsakos P (2003) Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). Evolutionary Computation 11(1):1–18

He Q, Wang L (2007) An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Engineering Applications of Artificial Intelligence 20(1):89–99

He S, Prempain E, Wu QH (2004) An improved particle swarm optimizer for mechanical design optimization problems. Engineering Optimization 36(5):585–605

Hu X, Eberhart RC, Shi Y (2003) Engineering optimization with particle swarm. In: 2003 IEEE Swarm Intelligence Symposium, pp 53–57

Kaveh A, Talatahari S (2009) Engineering optimization with hybrid particle swarm and ant colony optimization. Asian journal of civil engineering 10(6):611–628

Kaveh A, Talatahari S (2010) An improved ant colony optimization for constrained engineering design problems. Engineering Computations 27(1):155–182

Lemonge AC, Barbosa HJ, Borgesc CC, Silva FB (2010) Constrained optimization problems in mechanical engineering design using a real-coded steady-state genetic algorithm. Mecánica Computacional 3329(95):9287–9303

Mezura Montes E, Coello Coello CA, Landa-Becerra R (2003) Engineering optimization using simple evolutionary algorithm. In: 15th IEEE International Conference on Tools with Artificial Intelligence, pp 149–156

Mezura Montes E, Coello Coello CA, Velázquez-Reyes J, Muñoz-Dávila L (2007) Multiple trial vectors in differential evolution for engineering design. Engineering Optimization 39(5):567–589

Ragsdell KM, Phillips DT (1976) Optimal Design of a Class of Welded Structures Using Geometric Programming. Journal of Engineering for Industry 98(3):1021

Ray T, Liew KM (2003) Society and civilization: An optimization algorithm based on the simulation of social behavior. IEEE Transactions on Evolutionary Computation 7(4):386–396

Rocha AMA, Fernandes EM (2009) Hybridizing the electromagnetism-like algorithm with descent search for solving engineering design problems. International Journal of Computer Mathematics 86(10-11):1932–1946

Sandgren E (1988) Nonlinear integer and discrete programming in mechanical design. Proceedings of the ASME design technology conference

Scheibenpflug A, Beham A, Kommenda M, Karder J, Wagner S, Affenzeller M (2015) Simplifying Problem Definitions in the HeuristicLab Optimization Environment. In: Companion Publication of the 2015 Genetic and Evolutionary Computation Conference, pp 1101–1108

Schumer M, Steiglitz K (1968) Adaptive step size random search. IEEE Trans. Automat. Contr. 13(3):270–276

Tomassetti G (2010) A cost-effective algorithm for the solution of engineering problems with particle swarm optimization. Engineering Optimization 42(5):471–495

Townsend JC (2016) Golinski's Speed Reducer. http://www.eng.buffalo.edu/Research/MODEL/mdo.test.orig/class2prob4/descr.html

Wagner S, Kronberger G, Beham A, Kommenda M, Scheibenpflug A, Pitzer E, Vonolfen S, Kofler M, Winkler S, Dorfer V, Affenzeller M (2014) Architecture and Design of the HeuristicLab Optimization Environment. In: Advanced methods and applications in computational intelligence, 1st edition, vol 6. Springer, New York, pp 197–261

Zhang M, Luo W, Wang X (2008) Differential evolution with dynamic stochastic selection for constrained optimization. Information Sciences 178(15):3043–3074