FRIEDRICH-ALEXANDER
UNIVERSITÄT
ERLANGEN-NÜRNBERG

RECHTS- UND WIRTSCHAFTS-
WISSENSCHAFTLICHE FAKULTÄT

Global Optimization Algorithms with Application to Non-Life Insurance Problems

Ralf Kellner

Working Paper

Chair for Insurance Economics
Friedrich-Alexander-University of Erlangen-Nürnberg

Version: June 2012

GLOBAL OPTIMIZATION ALGORITHMS WITH APPLICATION TO NON-LIFE INSURANCE PROBLEMS

Ralf Kellner*

ABSTRACT

If optimization problems cannot be solved through analytical methods, global optimization algorithms constitute useful tools to derive numerical solutions. Global optimization methods are often applied to non-life insurance optimization problems. The aim of this paper is to analyze the application of four global optimization algorithms to insurance optimization problems and compare the results with respect to the algorithm's fitness, which reflects the ability to reach the highest/lowest objective function value, and the computational intensity. The insurance problems examined in the analysis include typical optimal decisions concerning the insurer's investment choice and risk management instruments. The results show that for the problems and methods considered within the simulation analysis, genetic optimization leads to the best objective function values, while particle swarm optimization exhibits the lowest computational intensity.

1. INTRODUCTION

In the past decades, many optimization algorithms have been developed with the purpose to numerically optimize functions for which analytical methods are not available. Most of these method's functionalities are derived from principles of nature and biology, such as evolution or swarm intelligence. Analyses comparing the performance of these algorithms for typical optimization problems exist for different fields of science, e.g. statistics, engineering or bioinformatics (see, e.g. Vesterstrøm and Thomson, 2004; Paterlini and Krink, 2006), in which results regarding the algorithms' performance differ for each field of science due to differences in optimization and objective functions. However, even though these algorithms are applied to several insurance optimization problems (always only one algorithm to a specified optimization problem), a comparison of global optimization algorithms has not been conducted in the field of non-life insurance. Thus, it seems to be an important topic to conduct such an analysis in the field of non-life insurance. Hence, the aim of this paper is to compare the performance of global optimization algorithms with respect to non-life insurance optimization problems.

* Ralf Kellner is at the Friedrich-Alexander-University (FAU) of Erlangen-Nuremberg, Chair for Insurance Economics, Lange Gasse 20, 90403 Nuremberg, Germany, Tel.: +49 911 5302884, ralf.kellner@wiso.uni-erlangen.de.

Early literature in the field of optimization of non-life insurance problems is often based on assumptions such as normality for risk processes to derive analytical solutions (see, e.g. Kahane and Nye, 1975; Cummins and Nye, 1981) and focuses on optimization problems in a mean-variance framework, simultaneously considering the asset and liability side. Kahane and Nye (1975) estimate parameters for the distributions of nineteen lines of business of an insurer and two asset classes as well as the correlation matrix for the risk process and follow a mean-variance approach to simultaneously optimize investment and underwriting decisions. As the approach applied in Kahane and Nye (1975) is very sensitive to parameter estimation, an alternative is applied in Kahane (1977) by using Sharpe's Single-Index Model, where each of the insurer's return on investment and underwriting decisions is explained through a relationship with a market performance index. This reduces the extent of parameter estimation, as instead of estimating the relationships between all of the insurer's underwriting and investment activities, only the relationship with the index has to be estimated for each activity. Cummins and Nye (1981) also derive mean-variance efficient frontiers for an underwriting and investment decision problem and determine how decision rules, such as ruin or utility theory, can be used to select mean-variance efficient points on the deduced frontier. Further examples for analyses concerning mean-variance efficient portfolio optimization of assets and liabilities are amongst others Krouse (1970) and Eisenberg and Kahane (1978).

Moreover, in the recent literature on non-life insurance, making simplifying assumptions, e.g. normality of risk processes, is usually avoided in order to capture the insurance optimization problems in a more realistic manner. Furthermore, the optimization problems' objective functions become more complex than mean-variance efficient problems and often include risk management instruments with complex payoff structures as well as tail risk measures within the constraints. As a consequence, analytical solutions are not easily available and solutions have to be analyzed through simulation analysis and numerical methods. Several applications of global optimization algorithms can be found in the non-life insurance literature. Zeng (2003, 2005) uses a genetic algorithm to search for the optimal quantity of industry loss warranty contracts, while Cummins, Lalonde and Phillips (2004) apply a hybrid genetic algorithm to solve an optimization problem searching for the optimal use of call spreads based on catastrophic loss indices. A pattern search method is employed by Yow and Sherris (2008), who optimize the firm value in a model allowing market imperfections, while Gatzert and Kellner (2011b) use differential evolution to search for the optimal usage of alternative risk transfer instruments in order to maximize the net shareholder value of a non-life insurer. In addition, Lei (2011) illustrates how genetic algorithms and Monte Carlo simulation can be used to derive solutions for cost minimization. Even though these methods are applied in the field of

non-life insurance, other popular global optimization algorithms, e.g. particle swarm optimization, have not been applied. Furthermore, global optimization algorithms are usually applied to the optimization problem without comparing their effectiveness to other global optimization methods.

However, with respect to other fields of science, the comparison of the global optimization algorithms' performance has been subject to several analyses. Storn and Price (1997) compare differential evolution to other evolutionary algorithms, annealing methods, and methods of stochastic differential equations within selected benchmark functions and detect differential evolution to be superior with respect to find the global optimum in a required number of function evaluations. Vesterstrøm and Thomson (2004) and Panduro et al. (2009) compare the performance of particle swarm optimization, genetic optimization and differential evolution. Vesterstrøm and Thomson (2004) observe that differential evolution outperforms particle swarm optimization and the genetic algorithm in 32 of 34 benchmark cases, while Panduro et al. (2009) find particle swarm optimization and differential evolution to be better than the genetic algorithm for an optimization problem in the field of electromagnetism. Moreover, differential evolution also proves to be more efficient in comparison to particle swarm optimization within an analysis by Paterlini and Krink (2006), examining the application to partitional clustering. While Pham and Wilamowski (2011) refine the Nelder-Mead simplex method and compare results of the newer to the original version, Bera and Mukherjee (2010) detect the latter to be inferior to a hybrid simulated annealing method for a multiple response optimization problem. However, Katari et al. (2007) detect the Nelder-Mead simplex method to be superior for certain cases of data clustering problems in comparison to an improved genetic algorithm.

Hence, in this paper, we expand previous work in the following way. We choose four global optimization algorithms, which are frequently used in the optimization literature as laid out above and apply them to various insurance optimization problems, which focus on optimal investment and risk management decisions. We first start with an optimization problem that is expressed in a mean-variance framework, analyzing an insurer's risk management decisions on the asset and liability side. This problem exhibits an analytical solution which can be compared to the solutions provided by the numerical methods of the global optimization algorithms. Furthermore, optimization problems maximizing the insurer's surplus or the net shareholder value under constraints accounting for the insurer's solvency situation are formulated. The optimization algorithm's results are compared with respect to the optimization algorithm's fitness, which describes the algorithm's ability to reach the highest/lowest objective

function value in case of a maximization/minimization problem (see, e.g. Paterlini and Krink, 2006) and the computational intensity. In addition, sensitivity analyses with respect to different input parameters of the global optimization algorithms and different starting values for the optimization algorithms are conducted.

Concerning the global optimization algorithms, we choose the Nelder-Mead simplex method, particle swarm optimization, differential evolution and genetic optimization. The Nelder-Mead simplex is one of the oldest optimization algorithms (see Nelder and Mead, 1965), but despite its age is still often applied (see, e.g. Price, Coope and Byatt, 2002; Katari et al., 2007). Due to the characteristic of having just one starting point instead of conducting a parallel search technique of multiple vectors, it exhibits advantages concerning the computational intensity, but also increases the likelihood of getting trapped in a local optimum. To avoid this potential drawback, modern optimization algorithms like particle swarm optimization, differential evolution and genetic optimization usually apply parallel search techniques. Within our analysis, our results show that genetic optimization leads to the highest objective function values, while particle swarm optimization exhibits the lowest computational intensity. Overall, our results show that the type of search technique (direct or parallel) and the number of heuristic rules that are applied during the optimization are crucial for the algorithms' performance.

The remainder of this paper is structured as follows. In Section 2, the optimization algorithms are presented, while Section 3 illustrates the optimization problems. Section 4 contains numerical analyses and Section 5 concludes.

2. Global Optimization Algorithms

In the following analysis, we compare the performance of four global optimization algorithms, the Nelder-Mead simplex method, differential evolution, genetic optimization and particle swarm optimization. The Nelder-Mead simplex is a simplex-based direct search method, while differential evolution and genetic optimization belong to the class of evolutionary algorithms, and particle swarm optimization follows the approach of swarm intelligence computation. While the Nelder-Mead simplex starts its search for a global optimum at one starting point, the remaining algorithms use sets of initial trial vectors and, thus, exhibiting a parallel search technique. All four optimization algorithms are applied to solve an optimization problem, which in general can be formulated as

$\min f(x),\ x \in \mathbb{R}^d,$

and, if necessary, be complemented with additional constraints

$$c = \begin{pmatrix} g_i(x) \le 0 \\ h_j(x) = 0 \end{pmatrix},\ i = 1,\ldots,n_{cg},\ \ j = 1,\ldots,n_{ch},$$

where x represents a vector with d parameters to be optimized, $f(x)$ the objective function to be minimized, $g_i(x)$ and $h_i(x)$ the constraint functions and n_{cg}, n_{ch} the number of constraints (see Marti, 2005). Depending on the characteristics and the progression of $f(x)$, different algorithms may lead to diverse results, which is based on different heuristic rules that are applied during the optimization.

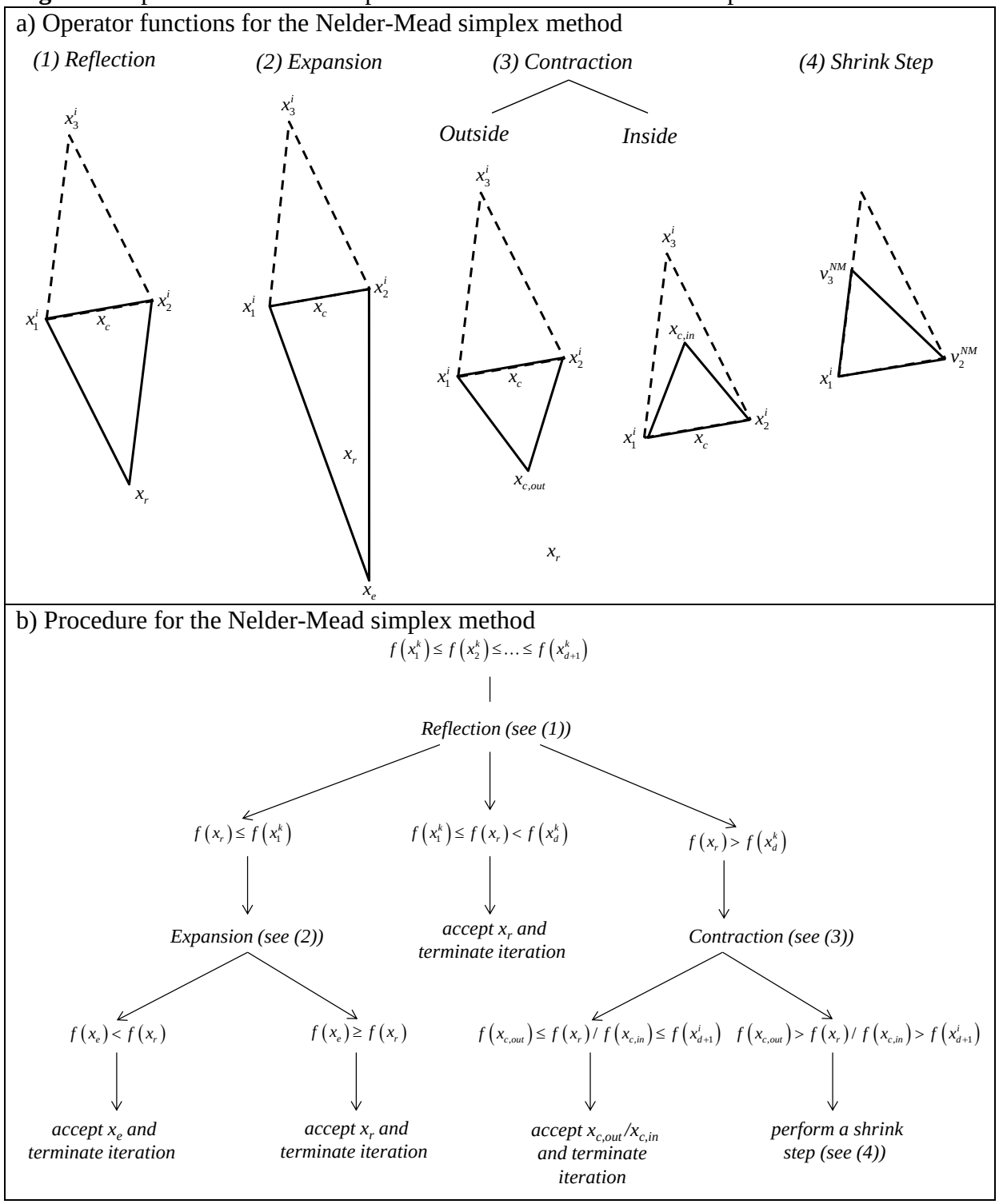
2.1 Nelder-Mead simplex method

The Nelder-Mead method was first published in 1965 (see Nelder and Mead, 1965) and is a method to optimize an unconstrained function $f(x)$ of d variables, $x \in \mathbb{R}^d$. To find an optimum, only function values are used. Thus, it represents a derivative-free method.

Each iteration step starts with a set of $d+1$ points, building a simplex,[1] which converts itself according to operation rules (see Lagarias et al., 1998). These rules are constructed to improve the best function's value at each iteration step or to stop if a stopping condition, e.g. the best function's value cannot be improved for more than a predefined amount of iteration steps, is reached. In particular, at the $i-th$ – iteration step $(i \ge 0)$, $d+1$ points, given through $d+1$ vertices from the simplex Δ_i, are *ordered* according to their function values $f(x)$ for $x \in \mathbb{R}^d$. Based on the order of the points $x_j,\ j = 1,\ldots,d+1$, the operations *reflection, expansion, contraction* and *shrink step* as illustrated in Figure 1a), which illustrates the four operations for a 2-dimensional problem, are conducted (see Lagarias et al., 1998).

[1] A simplex is a *d*-dimensional polytope, in which each simplex is defined by $d+1$ vertices, e.g. a simplex of dimension two is a triangle (see, e.g. Price, Coope and Byatt, 2002).

Figure 1: Operator functions and procedure for the Nelder-Mead simplex method



According to the order of function values $f\left(x_1^i\right) \le f\left(x_2^i\right) \le \ldots \le f\left(x_{d+1}^i\right)$, x_1^i denotes the best point, x_2^i the second best point and x_{d+1}^i the worst point, if the aim is to minimize f. After sorting the function values, reflection is applied, whereas the reflection point x_r is given through

$$x_r = x_c + \alpha^{NM} \cdot \left(x_c - x_{d+1}\right),$$

where x_c is the centroid[2] of the d best points (see Lagrias et al., 1998) and α^{NM} represents the reflection factor of the Nelder-Mead simplex method NM, which satisfies $\alpha^{NM} > 0$. The higher the value for α^{NM} is chosen, the further away moves the new reflection point from the centroid. Figure 1b) illustrates the three possibilities for the further development of the iteration. If the function value $f\left(x_r\right)$ of x_r lies between the best and the second worst function values $f\left(x_1^i\right) \le f\left(x_r\right) < f\left(x_d^i\right)$, x_r is accepted to replace x_{d+1}^i and the first iteration step is terminated. In case of $f\left(x_r\right) \le f\left(x_1^i\right)$, the expansion point x_e is calculated using

$$x_e = x_c + \gamma^{NM} \cdot \left(x_r - x_c\right),$$

with γ^{NM} as the expansion factor, satisfying $\gamma^{NM} > 1$ and $\gamma^{NM} > \alpha^{NM}$ (see, e.g. Price, Coope and Byatt, 2002). If the function value of x_e is higher or equal to the reflection point's function value, x_r is chosen to be the new vertex, otherwise x_e replaces x_{d+1}^i. For the third possible case $f\left(x_r\right) > f\left(x_d^i\right)$, an outside contraction is performed if $f\left(x_d^i\right) \le f\left(x_r\right) < f\left(x_{d+1}^i\right)$ and the outside contraction point $x_{c,out}$ is calculated according to

$$x_{c,out} = x_c + \beta^{NM} \cdot \left(x_r - x_c\right),$$

or an inside contraction is conducted, determining the inside contraction point $x_{c,in}$

$$x_{c,in} = x_c - \beta^{NM} \cdot \left(x_c - x_{d+1}^i\right),$$

if $f\left(x_r\right) \ge f\left(x_{d+1}^i\right)$, with β^{NM} representing the contraction factor, satisfying $0 < \beta^{NM} < 1$. If $f\left(x_{c,out}\right) \le f\left(x_r\right)$ or $f\left(x_{c,in}\right) < f\left(x_{d+1}\right)$, respectively, the contraction point $x_{c,out}$ or $x_{c,in}$, respectively, is accepted to be the new vertex for the next iteration step (see Lagrias et al., 1998). Otherwise, a shrink step is performed and the vertices for the simplex in the next iteration, $x_1, v_2^{NM}, \ldots, v_{d+1}^{NM}$, are calculated according to

---

[2] The centroid is the center of mass and in the example illustrated in Figure 1a) the point whose coordinates are the weighted means of the x_1^i and x_2^i coordinates, respectively.

$$v_i^{NM} = x_1 + \vartheta^{NM} \cdot \left(x_j - x_1 \right),$$

with ϑ^{NM} being the shrink step factor, satisfying $0 < \vartheta^{NM} < 1$. Summing up, each iteration step starts through reflecting the worst point at the centroid of the remaining points. If the function value of the new reflection point falls between the function values of the remaining point, the iteration step is closed. If the function value of the new reflection point leads to better results than the results of the remaining points, the search is extended through expansion, otherwise, if the function value of the reflection point is still worse than the function values of the remaining points, the search is narrowed down through contraction.

The algorithm explained above is repeated until a stopping criterion is satisfied, e.g. a maximum number of iterations is reached or the best objective function's value could not be improved for a predefined absolute or relative value. Accordingly, the algorithm must not lead to the global optimum, but can lead to results close to the global optimum in a lower computational time than, e.g. exhaustive testing in the search space. The method's quality depends on the choice of starting values, which must be set with the initialization of the method, operation factors $\left(\alpha^{NM}, \beta^{NM}, \gamma^{NM}, \vartheta^{NM} \right)$ and stopping conditions.

Due to its user-friendly application, the Nelder-Mead algorithm is frequently applied in the fields of statistics, engineering, physical and medical sciences (see Price, Coope and Bryatt, 2002). Nevertheless, a drawback of the Nelder-Mead algorithm is that its search for the global optimum could be trapped in a local optimum. This can be solved through trying different starting points for the algorithm. As an alternative, other optimization algorithms avoid this problem through simultaneously running several trial vectors. Examples are differential evolution, genetic optimization and particle swarm optimization, which are explained in detail in the next paragraphs.

2.2 Differential evolution

Differential evolution (DE) belongs to the class of evolutionary algorithms and is a parallel direct search method (see Storn and Price, 1997). It starts with a set of p d - dimensional trial vectors $x_{i,g}^{DE}$, $i = 1, 2, \ldots, p$, representing the population, and modifies its p trial vectors through the operation functions *mutation*, *crossover* and *selection* for a number of generations g, until a stopping condition is satisfied. Before the initialization, lower and upper bounds for each parameter, which should be used for the optimization, must be defined. The initial set of

trial vectors is then randomly drawn on a uniform probability space. Subsequently, for each trial vector $x_{i,g}^{DE}$, a mutant vector $v_{i,g}^{DE}$ is created through

$$v_{i,g}^{DE} = x_{r_0,g}^{DE} + F \cdot \left(x_{r_1,g}^{DE} - x_{r_2,g}^{DE} \right),$$

with random indices $r_0, r_1, r_2 \in \{1, 2, \ldots, p\}$, which have to be different to i, and F as the step size, satisfying $F > 0$ (see Price, Storn and Lampinen, 2006).[3] In the next step, an initial generation is created through crossing over the trial vector $x_{i,g}^{DE}$ and the mutant vector $v_{i,g}^{DE}$, whereas the trial vectors $u_{i,g}^{DE} = \left(u_{1i,g}^{DE}, u_{2i,g}^{DE}, \ldots, u_{di,g}^{DE} \right)$ are created according to

$$u_{ji,g}^{DE} = \begin{cases} v_{ji,g}^{DE} & \text{if } randb(j) \le CR \text{ or } j = rnbr(i) \\ x_{ji,g}^{DE} & \text{if } randb(j) > CR \text{ and } j \ne rnbr(i) \end{cases}, \quad j = 1, 2, \ldots, d,$$

where $CR \in [0,1]$ is the predefined crossover constant, $randb(j) \in [0,1]$ a uniform random number generator and $rnbr(i) \in 1, 2, \ldots, d$ a random index, ensuring that the trial vector $u_{i,g}^{DE}$ gets at least one parameter of the mutant vector $v_{i,g}^{DE}$ (see Storn and Price, 1997). The next generation of trial vectors $x_{i,g+1}^{DE}$ is then generated by comparing the objective function values of the initial vectors $f\left(x_{i,g}^{DE}\right)$ and the trial vectors $f\left(u_{i,g}^{DE}\right)$. Hence, $x_{i,g+1}^{DE}$ is created through

$$x_{i,g+1}^{DE} = \begin{cases} u_{i,g}^{DE} & \text{if } f\left(u_{i,g}^{DE}\right) \le f\left(x_{i,g}^{DE}\right) \\ x_{i,g}^{DE} & \text{otherwise} \end{cases},$$

(see Price, Storn and Lampinen, 2006). The process of mutation, crossover and selection is repeated until a stopping condition is satisfied. Through the parallel search technique, the distress of getting trapped in a local optimum is significantly reduced. However, at the same time this search technique increases the computational intensity in comparison to the Nelder-Mead simplex. While the mutation and crossover operators cause trial solutions to scan the objective function's surface, the selection operator memorizes good solutions from the previous generation.

2.3 Genetic optimization

The functionality of genetic optimization (GO) is derived from the mechanics of natural selections and natural genetics and proceeds over a number of generations g to reach the objec-

[3] As the index i has to be different to $r_0, r_1, r_2 \in \{1, 2, \ldots, p\}$, p must be greater or equal to four.

tive function's maximum (see Goldberg, 1989). Similar to differential evolution, a genetic algorithm starts with a set of d - dimensional initial vectors $x_{i,g}^{GO}$, $i = 1, 2, \ldots, p$, representing the population.

With each vector $x_{i,g}^{GO}$, the objective function's value $f\left(x_{i,g}^{GO}\right)$ is calculated. The population in the following generation g is created according to operator functions, which use data from the former generation to generate a new population and adopt rules to increase each generation's average objective function's value (see, e.g. Panduro et al., 2009). Thus, in contrast to differential evolution, the initial trial vectors are not passed to the next generation.[4] Operator functions among the genetic algorithm can differ (see, e.g. Filho, Alippi and Treleaven, 1994). In our analysis, we conduct a genetic algorithm based on Mebane and Sekhon (2011). The operator functions are displayed in Table 1. Even though the general procedure applied in differential evolution and genetic optimization is identical (parallel search of initial trial vectors, representing the population, is evolved over a number of generations through operator functions), results which are obtained may differ due to the different operator functions.

Especially the usage of an initial generation and the selection operation within differential evolution leads to new generations with identical vectors $x_{i,g}^{DE} = x_{i,g+1}^{DE}$, thus memorizing solutions from previous generations, whereas within the genetic algorithm, only a predefined number of vectors, which lead to the best objective function's values, is kept for the next generation through the cloning operation (see Mebane and Sekhon, 2011). For both methods, the choice of the population size p and the number of generations g is crucial for the optimization's quality. Nevertheless, due to the different operator functions, the optimal choices for p and g may also differ between differential evolution and genetic optimization. In addition, a tradeoff between a higher optimization's quality and a longer running time, coming along with an increase in p and g, exists.

[4] The cloning operator (see Table 1) constitutes an exception, as it copies a predefined amount of the best trial vector solutions from the previous generation into the next generation. Usually, only a few population members are generated through cloning, i.e. Mebane and Sekhon (2011) recommend to solely copy the current generation's best trial vector in the next generation.

Table 1: Operator functions for the genetic algorithm by Mebane and Sekhon (2011)

Cloning	Copy $x_{i,g}^{GO}$ in the next generation $x_{i,g+1}^{GO}$
Uniform mutation	Randomly choose $j = 1, 2, \ldots, d$. Select a value according to $x_{ji,g}^{GO,\sim} \sim U\left(x_{ji,g}^{GO,lower}, x_{ji,g}^{GO,upper}\right)$, where $x_{ji,g}^{GO,lower}$ and $x_{ji,g}^{GO,upper}$ represent the lower and upper bounds for the values of $x_{ji,g}^{GO}$. Set $x_{ji,g}^{GO,\sim} = x_{ji,g+1}^{GO}$.
Boundary mutation	Randomly choose $j = 1, 2, \ldots, d$. Set either $x_{ji,g+1}^{GO} = x_{ji,g}^{GO,lower}$ or $x_{ji,g+1}^{GO} = x_{ji,g}^{GO,upper}$, with probability 0.50 of using each value.
Non-uniform mutation	Randomly choose $j = 1, 2, \ldots, d$. Compute $q^{gen} = \left(1 - g^{current} / g^{\max}\right)^{B} u$, with $g^{current}$ representing the current generation number, $g^{\max}$ the maximum generation number, $B > 0$ a tuning parameter and $u \sim U(0,1)$. Set either $x_{ji,g+1}^{GO} = \left(1 - q^{gen}\right) \cdot x_{ji,g}^{GO} + q^{gen} \cdot x_{ji,g}^{GO,lower}$ or $x_{ji,g+1}^{GO} = \left(1 - q^{gen}\right) \cdot x_{ji,g}^{GO} + q^{gen} \cdot x_{ji,g}^{GO,upper}$ with probability 0.50 of using each value.
Polytope crossover	Using $m = \max(2, d)$ vectors $x_{i,g}^{GO}$ from the current population and m random numbers $s_l \in (0,1)$, so that $\sum_{l=1}^{m} s_l = 1$, set $x_{i,g+1}^{GO} = \sum_{l=1}^{m} s_l \cdot x_{l,g}^{GO}$.
Simple crossover	Pick a single integer j from d. Using two parameter vectors, $x_{1,g}^{GO}$ and $x_{2,g}^{GO}$, set $x_{j1,g+1}^{GO} = t \cdot x_{j1,g}^{GO} + (1-t) \cdot x_{j2,g}^{GO}$ and $x_{j2,g+1}^{GO} = t \cdot x_{j2,g}^{GO} + (1-t) \cdot x_{j1,g}^{GO}$, where $t \in (0,1)$ is a fixed number
Whole non-uniform mutation	Do non-uniform mutation for all the elements of $x_{ji,g+1}^{GO}$
Heuristic crossover	Pick $u \sim U(0,1)$. Using two parameter vectors, $x_{1,g}^{GO}$ and $x_{2,g}^{GO}$, compute $x_{1,g+1}^{GO} = u \cdot \left(x_{1,g}^{GO} - x_{2,g}^{GO}\right) + x_{1,g}^{GO}$. If $x_{1,g+1}^{GO}$ satisfies all constraints, it can be used, otherwise pick another u value and repeat until a preset number of attempts. If no $x_{1,g+1}^{GO}$ value is found, set $x_{1,g+1}^{GO}$ equal to the better value of $x_{1,g}^{GO}$ and $x_{2,g}^{GO}$. Two $x_{1,g+1}^{GO}$ values must be produced in this manner
Local-Minimum crossover	Run a BFGS[5] optimization with a vector $x_{1,g}^{GO}$ up to a preset number of iterations to produce $x_{i,g}^{GO,*}$. Pick $u \sim U(0,1)$ and calculate $x_{i,g+1}^{GO} = u \cdot x_{i,g}^{GO,*} + (1-u) \cdot x_{i,g}^{GO}$. If $x_{i,g+1}^{GO}$ satisfies all constraints, it can be used, otherwise shrink u by setting $u^{shrink} = u/2$ and recomputed $x_{i,g+1}^{GO}$ with u^{shrink}. If no satisfactory value for $x_{i,g+1}^{GO}$ can be found until a preset number of attempts is reached, return $x_{i,g}^{GO}$.

[5] BFGS refers to the Broyden-Fletcher-Goldfarb-Shanno quasi-Newton method. In the algorithm of Mebane and Sekhon (2011), the possibility exists to include and combine this method with the genetic algorithm. As our analysis intends to focus on the optimization through the genetic algorithm, we calibrate the algorithm such that no additional information derived through the BFGS method will be involved in the optimization process.

2.4 Particle swarm optimization

Particle swarm optimization (PSO) is a parallel direct search method, whose methodology is derived from the behavior of organisms such as bird flocking and fish schooling (see Panduro et al., 2009). At the beginning at time t, a set of particles $x_{i,t}^{PSO} = \left(x_{1i,t}^{PSO}, x_{2i,t}^{PSO}, \ldots, x_{di,t}^{PSO} \right)$, $i = 1, \ldots, s^{PSO}$ representing the swarm, is randomly placed in the search space while associated function values are determined.[6] Subsequently, the particles move around in the search space with velocity $v_{i,t}^{PSO} = \left(v_{1i,t}^{PSO}, v_{2i,t}^{PSO}, \ldots, v_{di,t}^{PSO} \right)$, where their movement depends on the history of their own and of the swarm members function values, i.e. over $t = 1, 2, \ldots, T$, the particles are gravitated in the directions of their own and the swarm's best function value achieved so far (see Poli, 2007). The velocity for each component in $t+1$, $v_{ji,t+1}^{PSO}$, $j = 1, 2, \ldots, d$, is calculated through

$$v_{ji,t+1}^{PSO} = w \cdot v_{ji,t}^{PSO} + c_1 \cdot u_1^r \cdot \left(x_{ji,t}^{PSO} \left(pbest \right) - x_{ji,t}^{PSO} \right) + c_2 \cdot u_2^r \cdot \left(x_{ji,t}^{PSO} \left(sbest \right) - x_{ji,t}^{PSO} \right),$$

where w is the inertia weight, c_1, c_2 are the acceleration constants, u_1^r, u_2^r are uniform distributed random numbers in $[0,1]$, $x_{ji,t}^{PSO} \left(pbest \right)$ is the $j-th$ component from the best solution, which is derived through the particle so far, $x_{ji,t}^{PSO} \left(sbest \right)$ represents the $j-th$ component from the best solution, which is derived through the swarm and $x_{ji,t}^{PSO}$ is the $j-th$ component of the current particle. Corresponding to the velocity in $t+1$, the new position of the particle's component is adjusted according to

$$x_{ji,t+1}^{PSO} = x_{ji,t}^{PSO} + v_{ji,t+1}^{PSO}.$$

After the movement of all particles, the first iteration step is closed. In the next step, this procedure is repeated until a stopping condition is satisfied. Thus, in contrast to differential evolution and genetic optimization, particle swarm optimization "memorizes" previous results through retaining all previous results for a possible solution to the optimization problem, while within differential evolution and genetic optimization, only results within the present generation are possible solutions for the optimization problem. On the one hand, this characteristic might enhance the computational intensity but on the other hand, it might also increase the probability of finding the global optimum. Furthermore, particle swarm optimization does not feature a set of different operator functions as they can be found for differential evolution

[6] At the beginning, the initial particles are comparable to the initial population members in differential evolution and genetic optimization.

and genetic optimization, which could adversely affect the search for the global optimum and decrease the computational intensity.

2.5 Comparison and evaluation

Summing up, the Nelder-Mead simplex method is the only algorithm that does not apply a parallel search technique, which reduces the computational intensity in comparison to the remaining algorithms applied in the analysis but enhances the probability to be trapped in a local optimum. Particle swarm optimization is similar to the genetic algorithm and differential evolution through applying a heuristic rule (determination of the velocity) to the particles in the swarm to improve the algorithm's results, but differs in that all previous solutions are memorized over the procedure, thus causing a high memory effort. The genetic algorithm creates a new set of vectors for each generation, solely containing a few of the previous generations' best trial solutions, while in differential evolution, vectors are retained, if they lead to better results than the new mutated trial vectors. Particle swarm optimization uses the particle personal's best result as well as the swarm's best result of the previous search to move new particles in the most promising area with respect to the best function's value. Otherwise, only the velocity is used to determine the evolvement of new trial solutions, whereas differential evolution and genetic optimization apply several operation functions.

To evaluate the optimization algorithm's quality within our analysis, two criteria are considered. In a first step, we examine which optimization algorithm leads to the best results, i.e. leads to the highest (lowest) value of the function to be maximized (minimized). The algorithm's ability to achieve the highest (lowest) objective function value is denoted as the optimization algorithm's *fitness* (see, e.g. Paterlini and Krink, 2006). Concerning the optimization methods applied in our analysis, an enhancement in the population and swarm size for differential evolution, genetic and particle swarm optimization, respectively, as well as in different starting values for the Nelder-Mead method increases the likelihood to find the highest (lowest) objective function value, while at the same time the computational intensity is raised. This tradeoff has to be kept in mind when analyzing the optimization methods efficiency. Therefore, in a second step, the *computational intensity* is examined by comparing the running time of each optimization run.

Besides the analysis of the algorithms' performance through the criteria fitness and computational intensity, an analysis is conducted which focuses on the impact of different random numbers for each of the algorithms' starting values. More precisely, the optimization algo-

rithms start with a random draw of an initial population (differential evolution and genetic optimization), swarm (particle swarm optimization) or starting point (Nelder-Mead simplex), which influences the optimization's procedure by the initial seeds that are used for this random draw, i.e. different values for the initial population, swarm or starting point might lead to different optimization results. To analyze the impact of a different initial generation, swarm or starting point in more depth, we analyze the optimization algorithm's robustness through the average relative deviation in modulus, which is determined according to

$$f(x)^{dev} = \frac{\sum_{i=1}^{n^{opt}} \left| \frac{f(x)^{origin} - f(x)_i^{diff.seed}}{f(x)^{origin}} \right|}{n^{opt}}, \; i = 1, \ldots, n^{opt}, \tag{1}$$

where, n^{opt} is the number of optimization runs with a different initial population, swarm or starting point, $f(x)^{origin}$ is the highest objective function value of a predefined standard case ('origin') and $f(x)_i^{diff.seed}$ is the result of the $i-th$ optimization run with different initial seeds in random number generation ('diff.seed').[7]

3. INSURANCE OPTIMIZATION PROBLEMS

To examine whether the optimization algorithms illustrated in Section 2 are suitable for optimizing insurance problems and to determine which algorithm is most efficient, we consider relevant optimization problems from the field of non-life insurance and alternative risk transfer, where each optimization problem intends to find optimal risk management strategies. The first is taken from Cummins and Song (2008) and exhibits a closed-form solution. This optimization problem is analyzed to obtain an impression how close the algorithms can get to the analytical optimum. The further optimization problems are formulated on the basis of Gatzert and Kellner (2011a) and Gatzert and Kellner (2011b), respectively, and do not exhibit closed-form solutions and thus require simulation techniques. The illustration of the optimization problems in this section is conducted with the purpose to clarify the optimizations' idea and notation. A detailed illustration of the model frameworks behind the optimization problems can be found in Cummins and Song (2008), Gatzert and Kellner (2011a), and Gatzert and Kellner (2011b).

[7] Cummins, Lalonde and Phillips (2004) apply a similar procedure to determine whether the optimization results are stable with respect to different intervals for starting values of the optimization.

3.1 Optimizing investment and reinsurance decisions

Considering a holistic risk management approach regarding the asset and liability side, a tradeoff between hedging decisions on both sides exists. If risky investment decisions on the asset side are made, higher expenses for risk management have to be spent and/or less risky underwriting decisions have to be done on the liability side to achieve a desired overall risk level, whereas less risky investment decisions on the asset side allow riskier underwriting decisions on the liability side. Cummins and Song (2008) analyze substitution effects between an insurer's investment decision and its risk management decision on the liability side. Therefore, they set up a mean-variance optimization problem that is optimized over the fraction invested in risk-free assets and the amount of a proportional (quota share) reinsurance contract.

In their model, in $t=0$, an insurer invests a fraction γ^{IRE} of its initial capital IC_0^{IRE} in risk-free assets, while the remaining fraction is invested in high-risk assets. The initial capital consists of shareholder's initial contribution EC_0, premium income $\pi^{S_1,IRE}$ for insuring possible losses at $t=1$ and premium expenses for proportional reinsurance $\pi^{re,IRE}$, which pays a fraction β^{IRE} of the insurer's loss.[8] Hence, the initial capital is given by $IC_0^{IRE} = EC_0 + \pi^{S_1,IRE} - \pi^{re,IRE}$.[9] The superscript IRE denotes the first optimization problem, which we denote as the investment and reinsurance optimization problem. In $t=1$, the insurance company faces possible losses S_1 and receives payments from the proportional reinsurance contract. High-risk assets are assumed to be normally distributed and independent from the insurer's loss, where μ_H and σ_H^2 represent the expected value and variance of the high-risk assets' return. In this setting, the insurer's task is to maximize the expected wealth $W^{IRE}\left(x^{IRE}\right)$ with

$$x^{IRE} = \begin{pmatrix} \gamma^{IRE} \\ \beta^{IRE} \end{pmatrix} \in \square^2,$$

[8] In Cummins and Song (2008), e.g. equity capital is denoted as internal wealth and high-risk assets are named risky assets. To retain a consistent notation with the subsequent optimization problems, expressions and notations are adapted.

[9] IRE in the superscript clarifies that the notation is not valid for the following optimization problems, the investment in risk-free assets in the first optimization problem is not the same as the investment in low-risk assets γ^{RBC} in Section 3.2 and Section 3.3, e.g. $\gamma^{RBC} \neq \gamma^{IRE}$. Contrariwise, notations without superscript are consistent for the three optimization problems.

which consists of the expected return on investments and underwriting operations $E\left(R\left(x^{IRE}\right)\right)$

$$E\left(R\left(x^{IRE}\right)\right)=\left(1-\gamma^{IRE}\right)\cdot IC_0^{IRE}\cdot\mu_H+\gamma^{IRE}\cdot IC_0^{IRE}\cdot r_f+\left(\delta^{S_1}-\beta^{IRE}\cdot\delta^{re,IRE}\right)\cdot E\left(S_1\right),$$

minus the variance of the insurer's position $\sigma^2\left(R\left(x^{IRE}\right)\right)$, given by

$$\sigma^2\left(R\left(x^{IRE}\right)\right)=\left(1-\gamma^{IRE}\right)^2\cdot\left(IC_0^{IRE}\right)^2\cdot\sigma_H^2+\left(1-\beta^{IRE}\right)^2\cdot\sigma^2\left(S_1\right),$$

weighted with a coefficient of risk aversion q^{IRE}, where δ^{S_1} denotes the loading charged by the insurer within its premium calculation, $\delta^{re,IRE}$ the reinsurance premium loading, $E\left(S_1\right)$ and $\sigma^2\left(S_1\right)$ the expected value and variance of the insurance company's losses. Hence, the optimization problem is expressed as

$$\max_{x^{IRE}} f\left(x^{IRE}\right)=W^{IRE,*}\left(x^{IRE}\right)=\max_{x^{IRE}}\left\{E\left(R\left(x^{IRE}\right)\right)-\frac{q^{IRE}}{2}\cdot\sigma^2\left(R\left(x^{IRE}\right)\right)\right\}, \tag{2}$$

such that the closed-form solution for the optimal fraction invested in risk-free assets $\gamma^{IRE,*}$ can be calculated through taking the first order derivative and setting the equation equal to zero, leading to

$$\gamma^{IRE,*}=1-\frac{\left(\mu_H-r_f\right)\cdot\frac{\sigma^2\left(S_1\right)}{\left[E\left(S_1\right)\right]^2}}{q^{IRE}\cdot\sigma_I^2\cdot\frac{\sigma^2\left(S_1\right)}{\left[E\left(S_1\right)\right]^2}\cdot\left[EC_0+\left(\delta^{S_1}-\delta^{re,IRE}\right)\cdot E\left(S_1\right)\right]+\sigma_I^2\cdot\left[\left(1+\delta^{re,IRE}\right)^2\cdot r_f+\left(1+\delta^{re,IRE}\right)\cdot\delta^{re,IRE}\right]},$$

while the optimal fraction of proportional reinsurance $\beta^{IRE,*}$ is given through

$$\beta^{IRE,*}=1-\frac{\left(1+\delta^{re,IRE}\right)\cdot r_f+\delta^{re,IRE}}{q^{IRE}\cdot E\left(S_1\right)\cdot\frac{\sigma^2\left(S_1\right)}{\left[E\left(S_1\right)\right]^2}},$$

(see Cummins and Song, 2008).

3.2 Maximizing an insurer's surplus under solvency constraints

For an insurance company, an important task is to generate new earnings and at the same time meet requirements concerning the solvency situation. Hence, we express optimization problems that try to maximize the difference between the insurer's assets and liabilities at time T under certain constraints with respect to the insurer's solvency situation. The optimization problems are based on the model framework presented in Gatzert and Kellner (2011a), in which the impact of non-linear dependencies on the basis risk of industry loss warranties is examined in the context of solvency capital requirements and the insurer's free surplus. Within the model framework, in $t=0$, the insurance company collects its premium income for insuring possible losses in $t=1$ and decides to purchase an industry loss warranty or an aggregate excess of loss reinsurance contract to manage its underwriting risk.[10] Furthermore, the initial capital, which consists of shareholder's initial contribution E_0, premium income $\pi^{S_1,SUP}$, which is calculated on the basis of the expected value for the insurer's loss and an additional constant loading, and premium expenses $\pi^{ILW,SUP}$, is invested in high- and low-risk assets. The superscript SUP denotes the surplus optimization problem, which is illustrated in this subsection. In $t=1$, the insurer faces possible losses S_1 and receives payments from the industry loss warranty contract $X_1^{ILW,SUP}$. The industry loss warranty's payoff is given by

$$X_1^{ILW,SUP}\left(A^{ILW,SUP},L^{ILW},Y\right)=\min\left(\max\left(S_1-A^{ILW,SUP},0\right),L^{ILW}\right)\cdot 1\{I_1>Y\},$$

where $A^{ILW,SUP}$ is the contract's attachment point, L^{ILW} the ILW's layer limit that constitutes the maximum payment and $1\{I_1>Y\}$ represents the trigger function, which is equal to 1, if the industry loss I_1 in $t=1$ exceeds a predefined trigger level Y and 0 otherwise (see Gatzert, Schmeiser and Toplek, 2011). The premium for the industry loss warranty contract is based on the expected value principle, such that its price is determined according to

$$\pi^{ILW,SUP}=E\left(X^{ILW,SUP}\right)\left(1+\delta^{ILW,SUP}\left(\rho_\tau\left(S_1,I_1\right)\right)\right).$$

Gatzert and Kellner (2011a) assume a constant loading $\delta^{ILW,SUP}$ for the industry loss warranty contract. In general, indices for index-linked instruments can be structured according to the insurer's needs on the basis of zip-codes (see, e.g. Major, 1999; Cummins, Lalonde and Phil-

[10] In our analysis, the insurance company exclusively purchases an industry loss warranty contract, such that the case without risk management or with an excess of loss reinsurance contract is not explained in detail in the following. Furthermore, see Gatzert and Schmeiser (2011) for a detailed discussion of industry loss warranty's key characteristics.

lips, 2004), which can be done with the purpose to decrease potential basis risk that in general decreases for higher values of $\rho_\tau(S_1, I_1)$. However, more individually structured indices are less transparent, which might increase the costs associated with these contracts (see, e.g. SwissRe, 2006). One way to integrate this behavior is to vary $\delta^{ILW,SUP}$ subject to Kendall's tau $\rho_\tau(S_1, I_1)$.[11] Hence, we assume that the costs, which are represented through $\delta^{ILW,SUP}\left(\rho_\tau(S_1, I_1)\right)$, are calculated through a convex cost function that depends on $\rho_\tau(S_1, I_1)$, and can be determined through

$$\delta^{ILW,SUP}\left(\rho_\tau(S_1, I_1)\right) = \rho_\tau^2(S_1, I_1).$$

The extension of the cost function represents a tradeoff between lower potential basis risk and higher costs due to a more individually structured index. The insurer's losses as well as the assets are assumed to be lognormal distributed, while the dependence structure among assets and liabilities is modeled through copulas (see Gatzert and Kellner, 2011a). Considering the optimization problem, the insurer's aim is to maximize the insurer's surplus in $t = 1$ with respect to the insurer's solvency situation. The surplus is given through

$$SUP_1\left(x^{SUP}\right) = \left(EC_0 + \pi^{S_1,SUP} - \pi^{ILW,SUP}\right) \cdot \left(\gamma^{SUP} \cdot e^{r_L} + \left(1 - \gamma^{SUP}\right) \cdot e^{r_H}\right) + X_1^{ILW,SUP} - S_1,$$

$$x^{SUP} = \begin{pmatrix} \gamma^{SUP} \\ A^{ILW,SUP} \\ \rho_\tau(S_1, I_1) \end{pmatrix} \in \mathbb{R}^3$$

where $A^{ILW,SUP}$ describes an ex ante chosen attachment point of the industry loss warranty contract,[12] EC_0 is the initial capital, γ^{SUP} denotes the fraction invested in low-risk assets and r_L / r_H are the normally distributed continuous one-period returns for low- (L) and high-risk assets (H). The solvency situation is represented through the shortfall probability SP_1 that can be determined according to

$$SP_1 = P\left(SUP_1\left(x^{SUP}\right) < 0\right).$$

[11] In practice, additional costs for the individually structured index might depend on the relationship between the protection buyer and seller, i.e. if a long term relationship between both counterparties exists, such that an individual index can be used for more than one transaction, it is likely that no additional costs occur for the individual index. Otherwise, if the transaction is conducted only once, individual indices might be more expensive.

[12] Further contract parameters like the layer limit or the ILW's trigger level can be chosen as well. In the following analysis only the attachment point will be varied within the optimization.

In general, the solvency situation can be accounted for in two different ways in regard to the optimization procedure. First, the objective function (here: $SUP_1\left(x^{SUP}\right)$) can be divided by the risk measure (here: SP_1), or, second, the solvency situation can be added to the constraints of the optimization problem, such that a very low value of the objective function is passed to the optimization algorithm if the constraint is broken (see Zeng, 2003). For our analysis, this leads to two different types of optimization functions. First,

$$\max_{x^{SUP}} f\left(x^{SUP}\right) = \frac{SUP_1^*\left(x^{SUP}\right)}{SP_1}, \tag{3}$$

subject to the constraints c_1^{SUP}, that the fraction invested in low-risk assets as well as Kendall's tau between the insurer's loss and the index have to be between zero and one and that the attachment point of the industry loss warranty contract $A^{ILW,SUP}$ is set to be in the range of a predefined barrier $\left[\underline{A}^{ILW,SUP}, \overline{A}^{ILW,SUP}\right]$

$$c_1^{SUP} = \begin{pmatrix} 0 \le \gamma^{SUP} \le 1 \\ 0 \le \rho_\tau\left(S_1, I_1\right) \le 1 \\ \underline{A}^{ILW,SUP} \le A^{ILW,SUP} \le \overline{A}^{ILW,SUP} \end{pmatrix},$$

and second,

$$\max_{x^{SUP}} f\left(x^{SUP}\right) = SUP_1^*\left(x^{SUP}\right), \tag{4}$$

subject to the constraints c_2^{SUP}, which extend c_1^{SUP} in a way that the shortfall probability is not allowed to exceed a certain predefined level $\overline{SP_1}$,

$$c_2^{SUP} = \begin{pmatrix} 0 \le \gamma^{SUP} \le 1 \\ 0 \le \rho_\tau\left(S_1, I_1\right) \le 1 \\ \underline{A}^{ILW,SUP} \le A^{ILW,SUP} \le \overline{A}^{ILW,SUP} \\ SP_1 \le \overline{SP_1} \end{pmatrix}.$$

In the numerical analysis in Section 4, different parameter combinations for the optimization problems according to Equation (3) and (4) are analyzed, i.e. first, we optimize over one of the variables $\left\{\gamma^{SUP}, A^{ILW,SUP}, \rho_\tau\left(S_1, I_1\right)\right\}$, while the remaining variables are kept constant, and, second we optimize over two variables, while the remaining variable is kept constant. Third,

the optimization is done over all three variables. Hence, seven different combinations for the optimization are taken into account, which overall leads to 14 different optimization problems with respect to the two optimization functions according to Equation (3) and (4). In particular, the combinations for each optimization problem are given through

$$\left\{\gamma^{SUP}, A^{ILW,SUP}, \rho_\tau(S_1, I_1), \left(\gamma^{SUP}, A^{ILW,SUP}\right), \left(\gamma^{SUP}, \rho_\tau(S_1, I_1)\right), \left(A^{ILW,SUP}, \rho_\tau(S_1, I_1)\right), \left(\gamma^{SUP}, A^{ILW,SUP}, \rho_\tau(S_1, I_1)\right)\right\}.$$

3.3 Maximizing shareholder value

Concerning the next type of optimization problem, we choose a problem, which is expressed from a shareholder's perspective. In Gatzert and Kellner (2011b) a risk management strategy combining index-linked and indemnity based risk management instruments is considered with respect to maximizing the net shareholder value under safety constraints. The index-linked instrument is given through a binary industry loss warranty contract, paying a fraction of a layer limit, if the index exceeds a predefined trigger, whereas gap insurance represents an indemnity based instrument and pays a fraction of the gap between the insurer's loss and the index-linked instrument's payment. In $t = 0$, the insurance company invests its initial capital, consisting of shareholder's initial contribution EC_0 plus premium income $\pi^{S_1,SHV}$ minus premium expenses $\pi^{i,SHV}$, in high- and low-risk assets, $i = ILW^{SHV}, Gap$, where ILW^{SHV} denotes the binary industry loss warranty contract and Gap the gap insurance contract. SHV in superscript denotes the optimization problem as the shareholder value optimization problem. In $t = 1$, the insurer has to pay for the insured losses and receives payments from risk management instruments. The payment of the binary loss warranty contract $X_1^{ILW,SHV}\left(\alpha^{SHV}\right)$ is given by

$$X_1^{ILW,SHV}\left(\alpha^{SHV}\right) = \alpha^{SHV} \cdot L^{ILW} \cdot 1\{I_1 > Y\},$$

whereas the gap insurance's payoff $X_1^{gap,SHV}\left(\beta^{SHV}\right)$ can be calculated according to

$$X_1^{gap,SHV}\left(\beta^{SHV}\right) = \max\left\{\beta^{SHV} \cdot \left(S_1 - X_1^{ILW,SHV}\left(\alpha^{SHV}\right)\right), 0\right\},$$

where α^{SHV} and β^{SHV} are the fractions of the industry loss warranty and the gap insurance contract, respectively. The net shareholder value is the today's value of equity capital in $t = 1$ minus the initial contribution of shareholders and should be maximized over the optimal fractions of risk management instruments. Hence, the optimization problem is expressed as

$$\max_{x^{SHV}} f\left(x^{SHV}\right) = SHV_0^*\left(x^{SHV}\right) =$$
$$= \max_{x^{SHV}} \left\{ \left(S_0 - DPO\left(x^{SHV}\right)\right) \cdot \delta^{S_1,SHV} - \sum_{i \in \{ILW,gap\}} \delta^{i,SHV} \cdot \left(e^{-r_f} \cdot E^Q\left(X_1^{i,SHV}\right)\right) \right\}, \tag{5}$$

$$x^{SHV} = \begin{pmatrix} \alpha^{SHV} \\ \beta^{SHV} \end{pmatrix} \in \mathbb{R}^2,$$

with $\left(S_0 - DPO\left(x^{SHV}\right)\right)$ representing the value of payments to policyholders at $t = 0$, DPO the default put option, which represents the loss in case of the insurer's default, $\delta^{S_1,SHV}$ the loading policyholder's are willing to pay, $\delta^{i,SHV}$ the loading demanded within the premium calculation for the risk management instruments and $\left(e^{-r_f} \cdot E^Q\left(X_1^{i,SHV}\right)\right)$ the discounted expected value of the risk managements payments under the risk-neutral pricing measure Q.[13] The optimization problem is solved subject to the constraints c^{SHV} that the insurer's shortfall probability is not allowed to exceed a predefined level $\overline{SP_1}$, the fractions α^{SHV} and β^{SHV} are in the range between zero and one and that the premium payment is consistent with the shortfall probability implied by the risk management strategy, as the loading, policyholders are willing to pay in $t = 0$, depends on the insurer's shortfall probability in $t = 1$. Thus, the shortfall probability, which is "promised" $\left(SP_0^{promised}\right)$ and applied to determine the insurer's premium income in $t = 0$ has to be equal to the shortfall probability, realized in $t = 1$,[14]

$$c^{SHV} = \begin{pmatrix} SP_1 \le \overline{SP_1} \\ 0 \le \alpha^{SHV} \le 1 \\ 0 \le \beta^{SHV} \le 1 \\ \pi^{S_1,SHV} \overset{!}{=} \left(S_0 - DPO\left(x^{SHV}\right)\right) \cdot \left(1 + \delta^{S_1,SHV}\right) \end{pmatrix}.$$

Analogously to the optimization problems given in Equation (3) and (4), the optimization is done for the variable combinations

$$\left\{\alpha^{SHV}, \beta^{SHV}, \left(\alpha^{SHV}, \beta^{SHV}\right)\right\}.$$

Thus, in total 18 different optimization problems are examined and compared in the numerical analysis in Section 4.

---

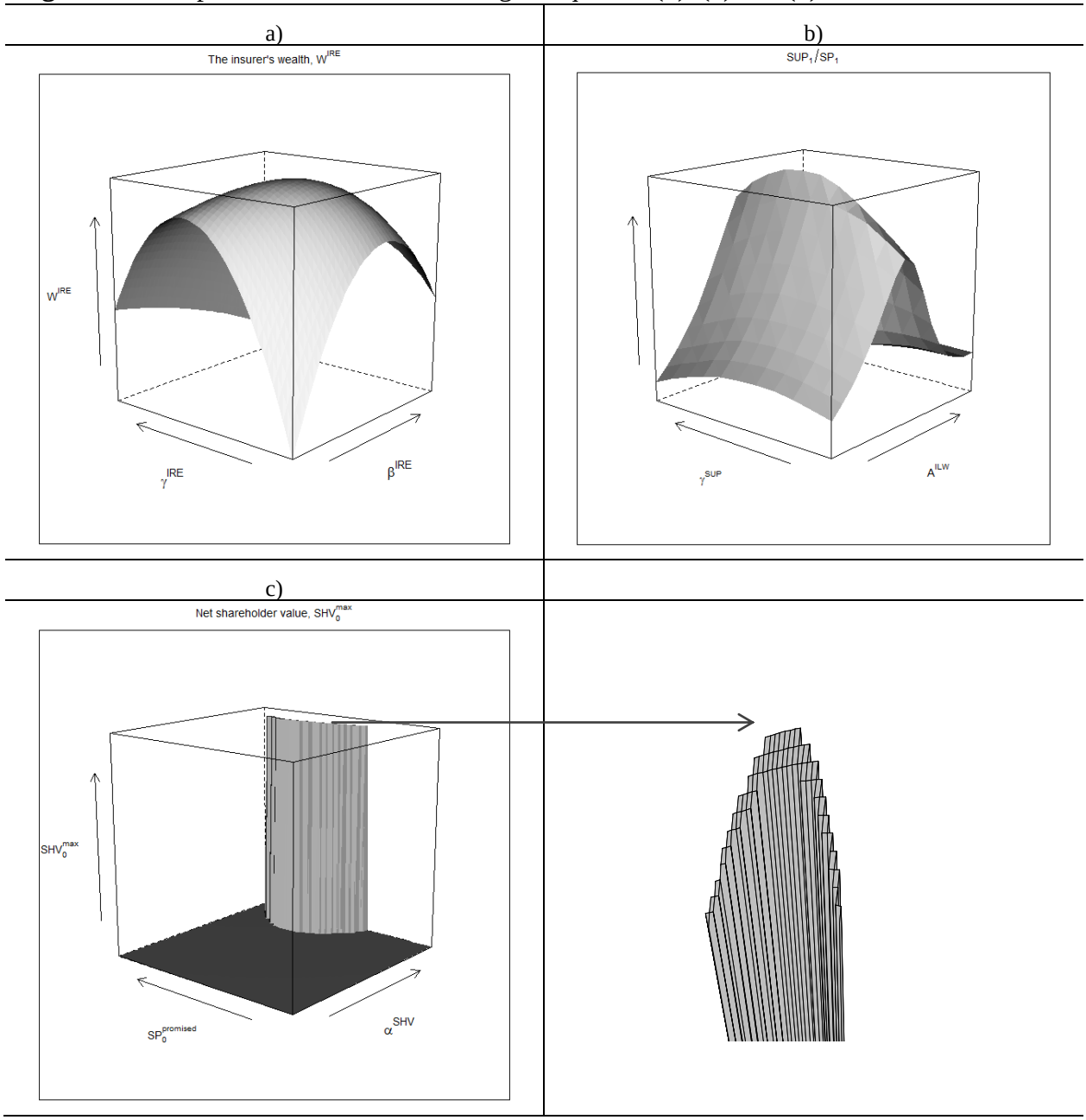
[13] See Gatzert and Kellner (2011b) for the derivation of Equation (5).

[14] For a detailed explanation, see Gatzert and Kellner (2011b).

3.4 Graphical illustration of the optimization problems

Figure 2 illustrates some examples for the objective functions described in this section. Figure 2a) and 2b) display the functions for the expected wealth W^{IRE} (see Equation (2)) and the insurer's surplus divided through the shortfall probability $SUP_1\left(x^{SUP}\right)/SP_1$ (see Equation (3)), dependent on $\left\{\gamma^{IRE},\beta^{IRE}\right\}$ and $\left\{A^{ILW,SUP},\gamma^{SUP}\right\}$.

Figure 2: Examples for Functions according to Equation (2), (3) and (5)



Furthermore, Figure 2c) displays the case for the net shareholder value according to Equation (5) in dependence of the industry loss warranty fraction α^{ILW} and the shortfall probability,

which is used for the premium income calculation $\left(SP_0^{promised}\right)$. In general, constraints are integrated in the optimization problems in two ways. First, bounds for the variables, which are used within the optimization, are enforced by calibrating the optimization, such that the initial trial solutions are only drawn in the permitted search space. Second, constraints, which affect quantities that are calculated during the optimization process, e.g. the shortfall probability, are incorporated in a way that in case of maximization (minimization), a very low (high) value of the objective function is passed to the optimization algorithm if the constraint is not satisfied (see, e.g. Mullen et al., 2011; Zeng, 2003). This can lead to a non-continuous surface, as it is the case for the objective function according to Equation (5), which is displayed in Figure 2c).

The algorithms illustrated in Section 2 usually are intended for minimization, whereas in our analysis, maximization problems are considered. Thus, each objective function is calibrated in a way that the negative function value is passed to the optimization algorithm. Moreover, further approaches for the optimization, e.g. robust optimization, can exist, which however, might be associated with the restructuring and different illustrations of Equations (2) – (5) and further assumptions concerning the optimizations' method. Furthermore, the analysis' focus lies on comparing global optimization algorithms.

4. Numerical Analysis

This section studies the application of the four optimization algorithms described in Section 2 – the Nelder-Mead simplex method, differential evolution, genetic optimization and particle swarm optimization – to the 18 optimization problems illustrated in Section 3. The analysis examines which algorithm is the most efficient when being applied to the optimization problems with respect to the algorithms' fitness and computational intensity. Furthermore, sensitivity analysis concerning the number of population (differential evolution and genetic optimization), swarm size (particle swarm optimization) and different starting points (Nelder-Mead simplex) as well as different initial populations, swarms or starting points for the optimization are conducted (see Section 2.5, Equation (1)).

4.1 Input parameters

The input data for the optimization problems, illustrated in Equations (3), (4) and (5), are given in Gatzert and Kellner (2011a) and Gatzert and Kellner (2011b).[15] For the optimization problem displayed in Equation (2), the same input parameters as in Gatzert and Kellner (2011a) are taken with the exception that the loading of the reinsurance contract is assumed to be 20%, $\delta^{re,IRE} = 0.20$, instead of 0%, and the risk aversion parameter, which is not necessary for the analysis in Gatzert and Kellner (2011a), equals $q^{IRE} = 0.01$. Furthermore, for the optimization problems according to Equation (3) and (4), the attachment point of the ILW has to be in the range of 0 and 300, $A^{ILW,SUP} \in [0,300]$ and for the optimization problems according to Equation (4) and (5) the upper limit for the shortfall probability is set to 5% (see Gatzert and Kellner, 2011b). Moreover, a Gauss copula is used to model the dependence structure for the optimization problems according to Equations (3), (4) and (5), thus, assuming linear dependencies among all relevant risk processes.

Table 2 displays the input parameters for each optimization algorithm. While the population or swarm size, respectively, for differential evolution, genetic optimization and particle swarm optimization is set to 150, we initialize the Nelder-Mead simplex method with 150 different starting values, which are drawn from a uniform distribution space. The number of generations is set to 200 for differential evolution and genetic optimization, whereas 2000 iteration steps are performed within the Nelder-Mead simplex method and particle swarm optimization. The numbers for swarm and population size, number of starting values, generations and iteration steps are chosen with the purpose to constitute comparable conditions for the analysis.

For the Nelder-Mead method, the reflection, expansion, contraction, and shrink step factors are adopted from Lagarias et al. (1998). A comparably small number (0.10) is chosen for the step size within differential evolution, as according to Price, Storn and Lampinen (2006), effective values are rarely higher than 1.0, even if all positive values are permitted. The crossover constant is chosen based on Mullen et al. (2011), while the acceleration constants for particle swarm optimization are fixed following Eberhardt and Shi (2001). In case of the first optimization problem (see Equation (2)), in which the analytical solution is available, we write a function to calculate the objective function values, using the statistical software R, and apply the optimization methods to the corresponding function.

[15] The input data for the corresponding reference contracts are given in Table 6 and Table 7 in the Appendix.

Table 2: Input parameters for the optimization algorithms

Nelder-Mead Simplex Method		
Number of starting points		150
Reflection factor	α^{NM}	1.00
Expansion factor	γ^{NM}	2.00
Contraction factor	β^{NM}	0.50
Shrink step factor	ϑ^{NM}	0.50
Number of iteration steps		2000
Differential Evolution		
Population size	p	150
Number of generations	g	200
Step size	F	0.10
Crossover constant	CR	0.50
Genetic Optimization		
Population size	p	150
Number of generations	g	200
Particle Swarm Optimization		
Swarm size	s^{pso}	150
Acceleration constants	c_1, c_2	2.0, 2.0
Number of iteration steps		2000

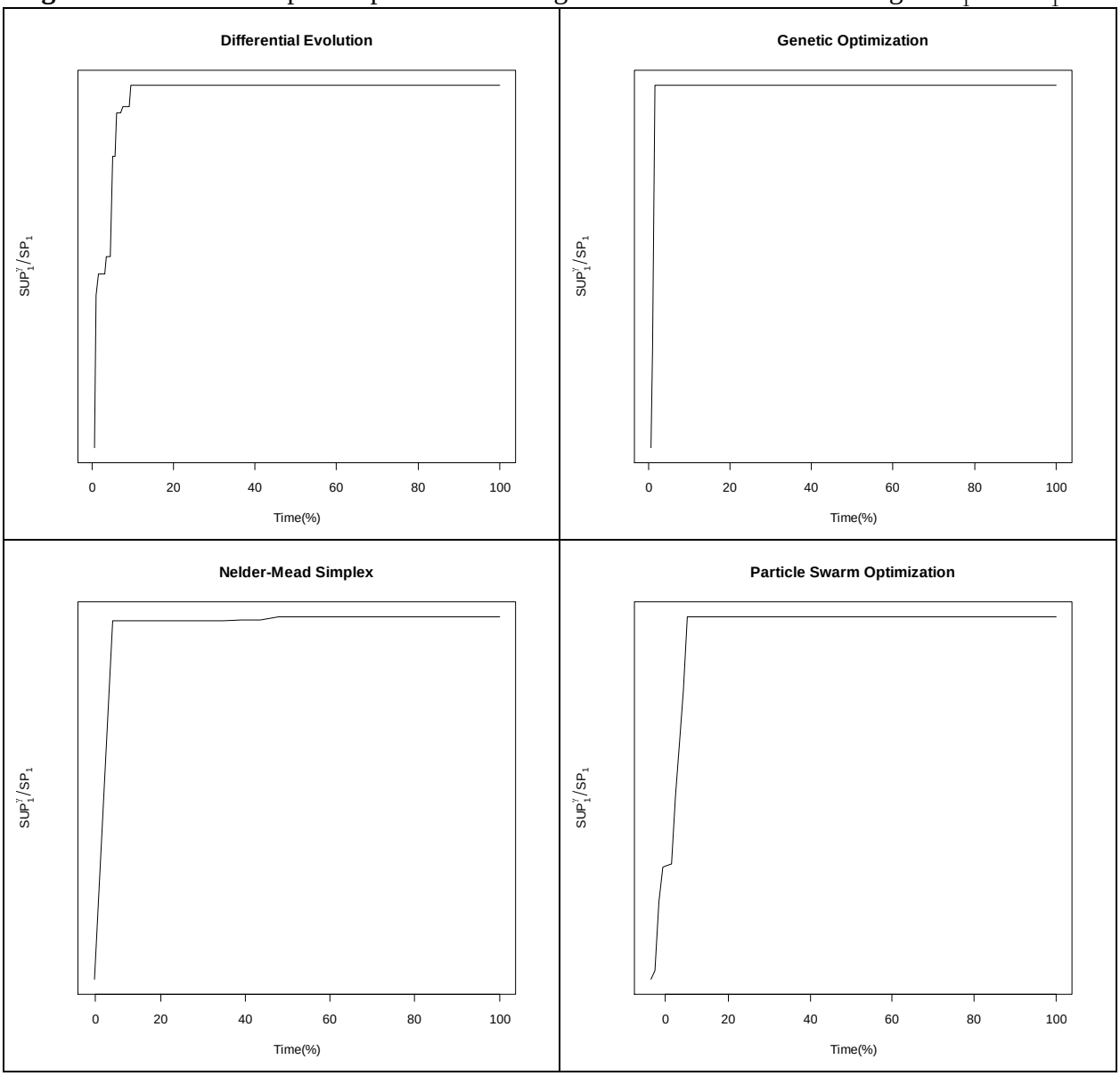
For the optimization problems according to Equation (3), (4) and (5), the numerical results are obtained using Monte Carlo simulation (see Glasserman, 2008). For each analysis, 200,000 sample paths are used and concerning the analysis of the optimization algorithm's robustness, 20 different sets of an initial population, swarm or starting points, respectively, are taken into account (see Section 2.5, Equation (1)). Each optimization run is calculated on an AMD Opteron Istanbul processor with 2.60 GHz.

4.2 Results

In a first step, we analyze performance profiles for the algorithms, which illustrate the development over time for the highest objective function values found during each optimization process, i.e. when the highest objective function value increases most during the genera-

tion/iteration steps. Therefore, besides the fixed number of generations or iterations, respectively, no further stopping conditions are included in the optimization procedure and each algorithm searches for the highest objective function value until the maximum number of generations or iterations is reached.

Figure 3: Performance profile plots for each algorithm in case of maximizing $SUP_1^{\gamma^{SUP}} / SP_1$



Note: The performance profile plots illustrate the development of the highest objective function values found during the time that is needed overall for the maximum number of iterations or generations.

Figure 3 exemplarily illustrates the performance plots for the maximization of $SUP_1^{\gamma^{SUP}} / SP_1$ dependent on γ^{RBC}.[16] Even though genetic optimization seems to be faster with respect to

[16] In the following, the variable(s) used for the optimization are given in superscript of the objective function.

finding the highest objective function values in comparison to the remaining algorithms, each of the optimization algorithms tends to find its highest objective function value before the maximum number of generations or iterations is reached.

By virtue of this observation, which can be found for each of the remaining optimization problems, we further include a stopping condition, such that the algorithms stop if the relative improvement of the objective function value is below 0.01%. This should lead to a decrease in computational intensity for each algorithm without lowering the optimizations accurateness.

Table 3 displays results for the optimization algorithms applied to the optimization problems, given in Equations (2), (3), (4) and (5) using 200,000 sample paths (except for the optimization problem according to Equation (2) which exhibits an analytical solution).[17] Given a certain optimization problem, the optimization algorithms are compared on the basis of the highest objective function and the running time.[18] The variables in superscript denote the variables that are used for the optimization, while the remaining variables are kept constant according to the numerical input parameters given in Section 4.1. Furthermore, considering the optimization problems according to Equations (3), (4) and (5), we optimize each problem for all possible variable combinations (see Section 3.2 and 3.3), e.g. for the optimization problem according to Equation (5), the optimization is done first for α^{SHV}, keeping β^{SHV} constant, second for β^{SHV}, keeping α^{SHV} constant and third for α^{SHV} and β^{SHV}. Each of the variable combination's function leads to different progressions for the objective functions, such that in summary 18 different optimization problems are analyzed (including the optimization problems according to the Equations (2), (3), (4) and (5)).

For the optimization problem according to Equation (2), a closed-form solution can be derived, leading to an optimal fraction invested in risk-free assets of -10.16% and an optimal proportional reinsurance contract reinsuring a fraction of 39.84%, which leads to the maximum expected wealth of 25.12 (see Section 3.1).[19] As can be seen in the right column of the upper row of Table 3, all optimization algorithms except particle swarm optimization attain

---

[17] A high number of sample paths is necessary to improve the simulation's robustness, e. g. for the optimization problem $SUP_1^{A^{SUP}} / SP_1$, the standard error equals 0.327, 0.293, 0.164, 0.133 and 0.115 for 25,000, 50,000, 100,000, 150,000 and 200,000 random numbers. In addition, latin hypercube sampling is applied within each simulation to further enhance the simulations' stability (see Glasserman, 2008).

[18] The highest objective function values and the shortest running time are are printed in boldface.

[19] No constraints are involved in the optimization, such that selling short is permitted.

the closed-form solution $\left(W^{IRE} = 25.12\right)$, which, in general, demonstrates that these algorithms are capable of reaching the global optimum.[20] While the Nelder-Mead simplex method, differential evolution and genetic optimization lead to identical solutions concerning the maximum expected wealth, particle swarm optimization derives a slightly lower value of $W^{IRE} = 25.11$ instead of 25.12. For each global optimization algorithm, the running time is below three seconds in case of the first optimization problem, which does not allow a comparison with respect to the computational intensity in that case.

Table 3 illustrates that genetic optimization leads to the highest objective function values for most of the optimization problems, but at the same time exhibits the highest computational intensity.[21] Contrary results can be found in case of particle swarm optimization, which leads to the lowest objective function values and the lowest computational intensity for most of the optimization problems. Furthermore, while differential evolution seems be most efficient after genetic optimization with respect to the algorithm's fitness, the Nelder-Mead simplex exhibits the lowest computational intensity after particle swarm optimization.

With respect to the algorithms' fitness, genetic optimization and differential evolution seem to be superior to the Nelder-Mead simplex, which emphasizes the advantageousness of the parallel search technique that is applied in case of the genetic optimization and differential evolution in comparison to the single starting point of the Nelder-Mead simplex. However, as particle swarm optimization also exhibits a parallel search technique and leads to lower objective function values than the Nelder-Mead simplex, the type of search technique might not be the only influence on the optimization algorithm's fitness. Furthermore, it seems to be the combination of the search technique and the number and types of operator functions, which influences the algorithm's fitness the most. In case of genetic optimization, eight different operator functions are applied, while differential evolution, the Nelder-Mead simplex and particle swarm optimization conduct three, four and one operator function, respectively.

---

[20] Further information would be necessary to make explicit specifications about the true global optimum. However, this is beyond the scope of this paper and the results given here are solely meant to illustrate how global optimization algorithms can perform in comparison to an analytical solution.

[21] Results for the optimization problem $SUP_1^{\gamma^{SUP}}$ are not displayed, as no value of γ^{SUP} can be found, such that the constraint of $SP_1 \leq 0.05$ is fulfilled.

Table 3: Results with respect to the optimization algorithms' fitness and computational intensity for all optimization problems (200,000 sample paths, except for the first optimization problem with analytical solution W^{IRE})

Obj. function	$\frac{SUP_1^{A^{SUP}}}{SP_1}$	$\frac{SUP_1^{\gamma^{SUP}}}{SP_1}$	$\frac{SUP_1^{\rho_\tau}}{SP_1}$	$\frac{SUP_1^{A^{SUP},\gamma^{SUP}}}{SP_1}$	$\frac{SUP_1^{A^{SUP},\rho_\tau}}{SP_1}$	$\frac{SUP_1^{\gamma^{SUP},\rho_\tau}}{SP_1}$	$\frac{SUP_1^{A^{SUP},\gamma^{SUP},\rho_\tau}}{SP_1}$	W^{IRE}		
Analytical sol.	-	-	-	-	-	-	-	25.120		
Differential Evolution	1784.247 (2) 0h, 46m, 51s (3)	**1447.575** (1) 0h, 31m, 37s (3)	1576.736 (2) 0h, 38m, 55s (3)	1792.484 (2) 0h, 39m, 57s (3)	2545.564 (2) 0h, 31m, 06s (3)	1575.345 (2) 0h, 54m, 54s (3)	2769.954 (2) 0h, 41m, 00s (3)	25.120 (1) <3seconds		
Genetic Optimization	**1784.729** (1) 0h, 55m, 32s (4)	1447.514 (2) 0h, 38m, 42s (4)	**1576.751** (1) 1h, 00m, 13s (4)	**1793.729** (1) 1h, 46m, 02s (4)	**2564.19** (1) 1h, 53m, 07s (4)	**1578.636** (1) 2h, 58m, 43s (4)	**2845.200** (1) 2h, 30m, 02s (4)	25.120 (1) <3seconds		
Nelder-Mead Simpplex	1784.198 (3) 0h, 14m, 37s (2)	1447.209 (3) 0h, 15m, 01s (2)	1576.391 (3) 0h, 16m, 19s (2)	1781.53 (3) 0h, 22m, 25s (2)	2506.422 (3) 0h, 23m, 59s (2)	1572.935 (3) 0h, 20m, 29s (2)	2648.007 (3) 0h, 35m, 21s (2)	25.120 (1) <3seconds		
Particle Swarm Optimization	1783.709 (4) **0h, 7m, 25s** (1)	1447.209 (3) **0h, 07m, 32s** (1)	1575.748 (4) **0h, 07m, 33s** (1)	1781.53 (3) **0h, 07m, 28s** (1)	2506.422 (3) **0h, 07m, 28s** (1)	1572.935 (3) **0h, 07m, 31s** (1)	2299.823 (4) **0h, 07m, 42s** (1)	25.110 (2) <3seconds		
Obj. function	$SUP_1^{A^{SUP}}$	$SUP_1^{\gamma^{SUP}}$	$SUP_1^{\rho_\tau}$	$SUP_1^{A^{SUP},\gamma^{SUP}}$	$SUP_1^{A^{SUP},\rho_\tau}$	$SUP_1^{\gamma^{SUP},\rho_\tau}$	$SUP_1^{A^{SUP},\gamma^{SUP},\rho_\tau}$	$SHV_0^{\alpha^{SHV}}$	$SHV_0^{\beta^{SHV}}$	$SHV_0^{\alpha^{SHV},\beta^{SHV}}$
Differential Evolution	87.044 (2) 0h, 46m, 52s (4)	- - -	**78.525** (1) 0h, 24m, 29s (3)	87.184 (3) 0h, 19m, 29s (2)	87.085 (2) 0h, 38m, 37s (3)	78.097 (3) 0h, 19m, 47s (2)	87.838 (2) 1h, 3m, 55s (3)	36.191 (3) 6h, 10m, 18s (3)	36.345 (2) 4h, 13m, 02s (3)	36.506 (2) 3h, 34m, 55s (3)
Genetic Optimization	**87.048** (1) 0h, 41m, 42s (3)	- - -	78.511 (2) 1h, 20m, 37s (4)	**87.759** (1) 1h, 39m, 59s (4)	**87.091** (1) 2h, 0m, 39s (4)	**78.634** (1) 1h, 13m, 07s (4)	**88.083** (1) 2h, 33m, 49s (4)	**36.301** (1) 11h, 19m, 12s (4)	**36.578** (1) 7h, 27m, 45s (4)	**36.552** (1) 15h, 18m, 53s (4)
Nelder-Mead Simplex	87.006 (3) 0h, 14m, 42s (2)	- - -	78.412 (3) 0h, 14m, 56s (2)	87.540 (2) 0h, 22m, 25s (3)	86.393 (3) 0h, 20m, 20s (2)	78.265 (2) 0h, 22m, 29s (3)	87.557 (3) 0h, 26m, 21s (2)	36.280 (2) 2h, 36m, 54s (2)	35.933 (3) 2h, 36m, 20s (2)	35.910 (3) 3h, 29m, 02s (2)
Particle Swarm Optimization	87.006 (3) **0h, 7m, 25s** (1)	- - -	78.412 (3) **0h, 05m, 45s** (1)	87.540 (2) **0h, 07m, 29s** (1)	86.352 (4) **0h, 07m, 28s** (1)	76.956 **(4)** **0h, 06m, 26s** (1)	87.557 (3) **0h, 07m, 45s** (1)	35.954 (4) **0h, 55m, 01s** (1)	35.933 (3) **0h, 54m, 29s** (1)	33.303 (4) **0h, 55m, 28s** (1)

In regard of the optimization problems considered in our analysis, the highest objective function values are achieved with a parallel search technique and the highest number and variety of operator functions. At the same time, this raises the computational intensity the most, which leads to a tradeoff that has to be taken into account when choosing an optimization algorithm for a certain optimization problem.

Each of the optimization problems depends on the initial population (differential evolution and genetic optimization), swarm (particle swarm optimization) or starting point (Nelder-Mead simplex), respectively, that are randomly drawn from the uniform probability space at the beginning of each optimization process. To determine the sensitivity of the results presented in Table 3 for different initial populations, swarms or starting points, we run each algorithm for a given optimization problem with 20 different sets of an initial population, swarm or starting points.

Table 4: Average relative deviations $f(x)^{dev}$ (see Equation (1)) for 20 different sets of an initial population, swarm or starting points for the optimization problems $SUP_1^{A^{SUP}} / SP_1$, $SUP_1^{A^{SUP},\gamma^{SUP}} / SP_1$ according to Equation (3) and $SUP_1^{A^{SUP}}$ and $SUP_1^{A^{SUP},\gamma^{SUP}}$ according to Equation (4)

Objective function	Differential Evolution	Genetic Optimization	Nelder-Mead Simplex	Particle Swarm Optimization
$\frac{SUP_1^{A^{SUP}}}{SP_1}$	0.01% (2)	0.00% (1)	0.02% (3)	0.11% (4)
$\frac{SUP_1^{A^{SUP},\gamma^{SUP}}}{SP_1}$	0.01% (2)	0.00% (1)	0.02% (3)	0.04% (4)
$SUP_1^{A^{SUP}}$	0.21% (2)	0.03% (1)	0.28% (3)	1.79% (4)
$SUP_1^{A^{SUP},\gamma^{SUP}}$	0.22% (2)	0.02% (1)	0.88% (3)	1.70% (4)

Table 4 exemplarily exhibits the average relative deviations (see Equation (1)) for the optimization problems $SUP_1^{A^{SUP}} / SP_1$, $SUP_1^{A^{SUP},\gamma^{SUP}} / SP_1$, $SUP_1^{A^{SUP}}$ and $SUP_1^{A^{SUP},\gamma^{SUP}}$, which are lowest for genetic optimization and highest for particle swarm optimization. This behavior can be observed for the remaining algorithms as well and is in line with the results, observed in Table 3, such that the algorithms with a parallel search technique and the highest number and variety of operator functions leads to the most stable optimization results. Furthermore, these results emphasize that different initial seeds for starting points, swarms or populations should be examined when applying optimization algorithms.

In the last step, sensitivity analyses with respect to the input parameters of the optimization algorithms are conducted. Each of the algorithms can be calibrated through various input parameters (e.g. the reflection factor for the Nelder-Mead simplex or the crossover constant CR for differential evolution), which differ depending on the algorithm that is applied (see Table 2). To keep the analysis comparable, we focus on different values for the population (differential evolution and genetic optimization), swarm size (particle swarm optimization) and number of different starting points (Nelder-Mead simplex). Varying further parameters, e.g. the step size within differential evolution, is not conducted as it cannot be determined which changes in parameter values of the remaining algorithms, e.g. the reflection factor for the Nelder-Mead simplex, would be necessary to create comparable conditions.

The increase in the number of population and swarm size as well as in different starting points in general increases the highest objective function values and/or reduces computational intensity as the solution which cannot be improved for a relative change of 0.01% might be found at an earlier point in time. Otherwise, the computational intensity per iteration step is enhanced as more possible solutions have to be simultaneously evaluated and an increase in the number of population and swarm size as well as in different starting points does not necessarily increase the highest objective function value. Table 5 exemplarily illustrates results for the optimization problems $SUP_1^{A^{SUP}} / SP_1$, $SUP_1^{A^{SUP},\gamma^{SUP}} / SP_1$, $SUP_1^{A^{SUP}}$ and $SUP_1^{A^{SUP},\gamma^{SUP}}$ for different values of population and swarm size as well as different starting points. For the majority of the optimization results, an increase in the number of population and swarm size and different starting points leads to higher objective function values and running times. Furthermore, for each value of population, swarm size and different starting points, similar results as in Table 3 can be observed such that the highest objective function values and running times occur for genetic optimization, while the lowest objective function values and running times can be found for particle swarm optimization. These results indicate that a change in the number of population and swarm size and different starting points influences each optimization algorithms performance in the same manner.

Summing up, our results show that in general, a tradeoff between the optimization algorithm's fitness and computational intensity occurs For the optimization problems considered in the analysis, the highest fitness can be found in case of genetic optimization, while the lowest computational intensity is observed for particle swarm optimization.

Table 5: Results for different population (diff. evolution and genetic optimization) and swarm (particle swarm optimization) sizes and starting points (Nelder-Mead simplex) for the optimization problems $SUP_1^{A^{SUP}} / SP_1$, $SUP_1^{A^{SUP},\gamma^{SUP}} / SP_1$, according to Equation (3) and $SUP_1^{A^{SUP}}$ and $SUP_1^{A^{SUP},\gamma^{SUP}}$ according to Equation (4)

	$\frac{SUP_1^{A^{SUP}}}{SP_1}$					$\frac{SUP_1^{A^{SUP},\gamma^{SUP}}}{SP_1}$			
Pop./Swarm Size/Starting Points	**Differential Evolution**	**Genetic Optimization**	**Nelder-Mead Simplex**	**Particle Swarm Optimization**	**Pop./Swarm Size/Starting Points**	**Differential Evolution**	**Genetic Optimization**	**Nelder-Mead Simplex**	**Particle Swarm Optimization**
50	1784.327 0h, 25m, 59s	1784.729 0h, 28m, 30s	1769.824 0h, 04m, 00s	1769.824 0h, 02m, 42s	50	1791.883 0h, 16m, 18s	1792.973 0h, 22m, 15s	1759.681 0h, 07m, 38s	1758.922 0h, 02m, 29s
100	1783.803 0h, 20m, 55s	1784.729 0h, 39m, 00s	1782.968 0h, 10m, 03s	1782.968 0h, 05m, 01s	100	1791.533 0h, 21m, 36s	1793.520 0h, 27m, 15s	1759.681 0h, 15m, 17s	1758.922 0h, 05m, 04s
150	1784.247 0h, 46m, 51s	1784.729 0h, 55m, 32s	1784.198 0h, 14m, 37s	1783.709 0h, 07m, 25s	150	1792.484 0h, 39m, 57s	1793.729 1h, 46m, 02s	1781.530 0h, 22m, 25s	1781.530 0h, 07m, 28s
200	1784.670 1h, 23m, 01s	1784.729 1h, 16m, 40s	1784.198 0h, 19m, 56s	1784.081 0h, 10m, 05s	200	1791.533 0h, 43m, 10s	1793.073 0h, 49m, 14s	1786.310 0h, 30m, 49s	1786.310 0h, 10m, 06s
250	1784.270 1h, 6m, 45s	1784.729 0h, 57m, 02s	1784.198 0h, 24m, 15s	1784.081 0h, 12m, 43s	250	1791.533 0h, 53m, 48s	1793.837 1h, 0m, 19s	1786.310 0h, 38m, 16s	1786.310 0h, 12m, 36s
	$SUP_1^{A^{SUP}}$					$SUP_1^{A^{SUP},\gamma^{SUP}}$			
Pop./Swarm Size/Starting Points	**Differential Evolution**	**Genetic Optimization**	**Nelder-Mead Simplex**	**Particle Swarm Optimization**	**Pop./Swarm Size/Starting Points**	**Differential Evolution**	**Genetic Optimization**	**Nelder-Mead Simplex**	**Particle Swarm Optimization**
50	87.040 0h, 14m, 29s	87.048 0h, 37m, 25s	87.006 0h, 04m, 00s	87.006 0h, 02m, 31s	50	87.721 0h, 36m, 35s	87.646 0h, 20m, 58s	86.180 0h, 07m, 38s	85.396 0h, 02m, 30s
100	87.047 0h, 48m, 28s	87.048 0h, 38m, 48s	87.006 0h, 10m, 03s	87.006 0h, 05m, 02s	100	87.587 0h, 52m, 11s	87.734 0h, 21m, 38s	86.180 0h, 15m, 29s	85.396 0h, 05m, 01s
150	87.044 0h, 46m, 52s	87.048 0h, 41m, 42s	87.006 0h, 14m, 42s	87.006 0h, 07m, 25s	150	87.184 0h, 19m, 29s	87.759 1h, 39m, 59s	87.540 0h, 22m, 25s	87.540 0h, 07m, 29s
200	87.038 0h, 51m, 44s	87.048 0h, 58m, 32s	87.006 0h, 19m, 45s	87.006 0h, 10m, 00s	200	87.754 2h, 05m, 07s	87.810 1h, 29m, 22s	87.540 0h, 30m, 34s	87.540 0h, 10m, 02s
250	87.045 1h, 18m, 03s	87.048 0h, 56m, 16s	87.027 0h, 23m, 21s	87.027 0h, 12m, 30s	250	87.447 0h, 53m, 12s	87.779 0h, 51m, 52s	87.540 0h, 38m, 12s	87.540 0h, 16m, 36s

5. CONCLUSION

This paper examined the application of global optimization algorithms to 18 non-life insurance optimization problems. We applied the Nelder-Mead simplex method, differential evolution, genetic optimization and particle swarm optimization, while the optimizations' results were evaluated with respect to the optimization's fitness, which refers to the algorithm's ability to derive the optimal objective function value, and computational intensity. Furthermore, sensitivity analyses concerning the number of population (differential evolution and genetic optimization), swarm size (particle swarm optimization) and different starting points (Nelder-Mead simplex) as well as different initial populations, swarms or starting points, for the optimization were conducted.

The optimization algorithms under consideration belonged to the family of direct search methods, but differed in various ways. The Nelder-Mead simplex method, in contrast to the remaining algorithms applied, did not exhibit a parallel search technique, thus, starting from one point. Particle swarm optimization evolved its optimal solution over particle trial vectors, which memorized previous trial solutions. Differential evolution and genetic optimization derived their optimal solutions over generations, in which only a few (in case of the cloning operator within genetic optimization) and a part of previous trial solutions (in case of the selection operator within differential evolution) were memorized. Contrariwise, differential evolution and genetic optimization applied several operation functions to find better solutions, while solely the velocity of particles determined the solution's devlopment within particle swarm optimization.

Concerning the insurance optimization problems, one objective function with a closed-form solution and no constraints, as well as several objective functions without closed-form solutions containing several constraints were chosen. The numerical results showed that, considering the investment and reinsurance optimization problem (exhibiting the closed-form solution), each of the global optimization algorithms, except particle swarm optimization, reached the closed-form solution, thus demonstrating their capability to reach or get close to the "true" global optimum. Furthermore, for the remaining optimization problems, genetic optimization led to the highest objective function values, while it exhibited the highest computational intensity in most of the cases. While oppositional results were found for particle swarm optimization, differential evolution was most efficient with respect to the algorithm's fitness after genetic optimization, while the Nelder-Mead simplex exhibited the lowest computational intensity, following particle swarm optimization. Furthermore, genetic optimization exhibited

the lowest value for the average relative deviations, if the algorithms were started with different initial populations, swarms or starting points. In a last step, the impact of the number of population, swarm size and different starting points was analyzed. The results showed that the number of population and swarm size as well as different starting points played an important role for each of the algorithms' fitness and computational intensity, but did not affect the general results for the optimization problems considered, such that genetic optimization led to the highest objective function values and particle swarm optimization to the lowest computational intensity for each value of the number of population and swarm size as well as different starting points.

In summary, the analysis showed that for the problems considered, genetic optimization leads to the highest fitness, while particle swarm optimization exhibits the lowest computational intensity. However, our results showed that the type of search method (parallel or direct) and the number of operator functions play a crucial role with respect to the optimization algorithms' fitness and computational intensity. Overall, an increase in the number of population and swarm size as well as starting points and in the number and variety of operator functions raises the optimization algorithms' fitness but also its computational intensity, a tradeoff that has to be kept in mind and evaluated individually when choosing a respective optimization algorithm.

REFERENCES

Bera, S., Mukherjee, I. (2010): Performance Analysis of Nelder-Mead and a Hybrid Simulated Annealing for Multiple Response Quality Characteristic Optimization. *Proceedings of the International Multi Conference of Engineers and Computer Scientists 2010* III, 1728–1732.

Cummins, J.D., Song, Q. F. (2008): Hedge the Hedgers: Usage of Reinsurance and Derivatives by PC Insurance Companies. *Working Paper, Wharton School, University of Pennsylvania, Philadelphia.*

Cummins, J. D., Lalonde, D., Phillips, R. D. (2004): The Basis Risk of Catastrophic-Loss Index Securities. *Journal of Financial Economics* 71(1), 77–111.

Cummins, J. D., Nye, D. J. (1981): Portfolio Optimization Models for Property-Liability Insurance Companies: An Analysis and Some Extensions. *Management Science* 27(4), 414–430.

Eberhardt, R. Shi, Y. (2001): Particle Swarm Optimization: Developments, Applications and Resources. *Proceedings of IEEE International Congress on Evolutionary Computation* 1, 81–86.

Eisenberg, S., Kahane, Y. (1978): An Analytical Approach to Balance Sheet Optimization and Leverage Problems of a Property-Liability Insurance Company. *Scandinavian Actuarial Journal* 4, 205–210.

Filho, J. R., Alippi, C., Treleaven, P. (1994): Genetic Algorithm Programming Environments. *Computer* 27(6), 28–43.

Gatzert, N., Kellner, R. (2011a): The Influence of Non-Linear Dependencies on the Basis Risk of Industry Loss Warranties. *Insurance: Mathematics and Economics* 49(1), 132–144.

Gatzert, N., Kellner, R. (2011b): The Effectiveness of Gap Insurance with Respect to Basis Risk in a Shareholder Value Maximization Setting. *Working Paper, Friedrich-Alexander-University (FAU) of Erlangen-Nürnberg*.

Gatzert, N., Schmeiser, H. (2011): Industry Loss Warranties: Contract Features, Pricing, and Central Demand Factors. *Journal of Risk Finance* 13(1), 13–31.

Gatzert, N., Schmeiser, H., Toplek, D. (2011): An Analysis of Pricing and Basis Risk for Industry Loss Warranties. *Zeitschrift für die gesamte Versicherungswissenschaft* 100(4), 517–537.

Glasserman, P. (2008): Monte Carlo Methods in Financial Engineering (New York: Springer Press New York, Inc.).

Goldberg, D. E. (1989): Genetic Algorithms in Search, Optimization and Machine (Boston, Addison-Wesley Publishing Company, Inc.).

Kahane, Y., Nye, D. J. (1975): A Portfolio Approach to the Property-Liability Insurance Industry. *Journal of Risk and Insurance* 42(4), 579–598.

Kahane, Y. (1977): Determination of the Product Mix and the Business Policy of an Insurance Company – A Portfolio Approach. *Management Science* 23(10), 1060–1069.

Katari, V., Satapathy, S. C., Murthy, J., Reddy, P. (2007). Hybridized Improved Genetic Algorithm with Variable Length Chromosome for Image Clustering. *International Journal of Computer Science and Network Security* 7(11), 121–131.

Krouse, C. G. (1970): Portfolio Corporate Assets and Liabilities with Special Application to Insurance. *Journal of Financial and Quantitative Analysis* 5(1), 77–104.

Lagarias, J. C., Reeds, J. A., Wright, M. H., Wright, P. E. (1998): Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions. *SIAM Journal on Optimization* 9(1), 112–147.

Lei, Y. (2011): Minimizing the Cost of Risk with Simulation Optimization Technique. *Risk Management and Insurance Review* 14(1), 121–144.

Major, J. A. (1999): Index Hedge Performance: Insurer Market Penetration and Basis Risk. In Kenneth A. Froot, ed., *The Financing of Catastrophe Risk* (Chicago: University of Chicago Press).

Marti, K. (2005): Stochastic Optimization Methods (Heidelberg, Springer Berlin).

Mebane, W. R. Jr., Sekhon, J. S. (2011): Genetic Optimization Using Derivatives: The rgenoud package for R. *Journal of Statistical Software* 42(11), 1–26.

Mullen, K. M., Ardia, D., Gil, D., Windover, D., Cline, J. (2011): DEoptim: An R package for Global Optimization by Differential Evolution. *Journal of Statistical Software* 40(6), 1–26.

Nelder, J. A., Mead, R. (1965): A Simplex Method for Function Minimization. *The Computer Journal* 7, 308–313.

Paterlini, S. Krink, T. (2006): Differential Evolution and Particle Swarm Optimization in Partitional Clustering. *Computational Statistics and Data Analysis* 50(5), 1220–1247.

Panduro, M. A., Brizuela, C. A., Balderas, L. I., Acosta, D. A. (2009): A Comparison of Genetic Algorithms, Particle Swarm Optimization and the Differential Evolution Method for the Design of Scannable Circular Antenna Arrays. *Progress in Electromagnetics Research B* 13, 171–186.

Pham, N., Wilamowski, B. M. (2011): Improved Nelder-Mead's Simplex Method and Applications. *Journal of Computing* 3(3), 55–63.

Poli, R (2007): Analysis of the Publications on the Applications of Particle Swarm Optimization. *Journal of Artificial Evolution and Application* 2008(1), 1–10.

Price, C. J., Coope, I.D., Byatt, D. (2002): A Convergent Variant of the Nelder-Mead Algorithm. *Journal of Optimization Theory and Applications* 113(1), 5–19.

Price, K., Storn, R., Lampinen, J. (2006): Differential Evolution – A Practical Approach to Global Optimization (Berlin – Heidelberg: Springer – Verlag).

Storn, R., Price, K. (1997): Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization* 11(4), 341–359.

SwissRe (2006): Securitization – New Opportunities for Investors and Insurers. Sigma No. 7/2006. Zürich: Swiss Reinsurance Company. Available at: http://www.swissre.com.

Vesterstrøm, J, Thomson, R. (2004): A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems. *Evolutionary Computation* 2, 1980–1987.

Yow, S., Sherris, M. (2008): Enterprise Risk Management, Insurer Value Maximization, and Market Frictions. *ASTIN Bulletin* 38(1), 293–339.

Zeng, L. (2003): Hedging Catastrophe Risk Using Index-Based Reinsurance Instruments. *Casualty Actuarial Society Forum* Spring, 245–268.

Zeng, L. (2005): Enhancing Reinsurance Efficiency using Index-Based Instruments. *Journal of Risk Finance* 6(1), 6–16.

APPENDIX

Table 6: Input parameters for the reference contract given in Gatzert and Kellner (2011a)

Available equity capital at time 0	E_0	\$40 million
Expected value and standard deviation of company loss (lognormally distributed)	$E(S_1), \sigma(S_1)$	\$117 million, \$66 million
Expected value and standard deviation of industry index (lognormally distributed)	$E(I_1), \sigma(I_1)$	\$1,450 million, \$3,550 million
Expected value and standard deviation for the return of high-risk assets (normally distributed)	μ_H, σ_H	8%, 20%
Expected value and standard deviation for the return of low-risk assets (normally distributed)	μ_L, σ_L	5.5%, 6.5%
Risk-free interest rate	r_f	2%
Investment in low-risk assets	γ	60%
Kendall's tau for low-risk and high-risk assets	$\rho_\tau(A_{1,L}, A_{1,H})$	0.2
Kendall's tau for company and index losses	$\rho_\tau(S_1, I_1)$	0.6
Kendall's tau for assets and liabilities	$\rho_\tau(A_1, L_1)$	0.1
Premium loading insurance contract	δ^{S_1}	30%
Premium loading reinsurance contract	δ^{re}	0%
Premium loading ILW	δ^{ILW}	0%
Layer limit for ILW and reinsurance contract	L^{ILW}, L^{re}	\$200 million
Industry loss trigger	Y	\$3,000 million
Attachment of the company's loss for ILW	A^{ILW}	\$100 million
Attachment of the company's loss for reinsurance	A^{re}	\$100 million

Table 7: Input parameters for the reference contract given in Gatzert and Kellner (2011b)

Available equity capital at time 0	E_0	\$48 million
Expected value of the company loss	$E(S_1)$	\$117 million
Expected value of the industry index	$E(I_1)$	\$1450 million
Drift and volatility of the company loss	μ_{S_1}, σ_{S_1}	0.025, 0.53
Drift and volatility of the industry index	μ_{I_1}, σ_{I_1}	0.025, 1.39
Drift and volatility of high-risk assets	$\mu_{A_{1,high}}, \sigma_{A_{1,high}}$	0.10, 0.20
Drift and volatility of low-risk assets	$\mu_{A_{1,low}}, \sigma_{A_{1,low}}$	0.57, 0.065
Risk-free interest rate	r_f	2%
Investment in high risk investment	γ	25%
Policyholder's risk sensitivity	q	5
Kendall's tau for low risk and high risk investment	$\rho_\tau(A_{1,high}, A_{1,low})$	0.20
Kendall's tau for company and index losses	$\rho_\tau(S_1, I_1)$	0.70
Kendall's tau for assets and company losses	$\rho_\tau(A_1, S_1)$	0.10
Kendall's tau for assets and index losses	$\rho_\tau(A_1, I_1)$	0.10
Premium loading for an insurer without default risk	$\delta^{\max,S_1}$	40%
Premium loading ILW	δ^{ILW}	5%
Premium loading gap insurance and proportional reinsurance	$\delta^{gap}, \delta^{re}$	20%
Maximum shortfall probability	$\overline{SP_1}$	5%
Layer limit for ILW	L^{ILW}	\$200 million
Industry loss trigger	Y	\$2,000 million