A HYBRID GENETIC-QUANTITATIVE METHOD FOR RISK-RETURN OPTIMISATION OF CREDIT PORTFOLIOS

Frank Schlottmann[1]
Detlef Seese[2]

Institute AIFB
University Karlsruhe

Abstract

This paper proposes a new combination of quantitative models and Genetic Algorithms for the task of optimising credit portfolios. Currently, quantitative portfolio credit risk models are used to calculate portfolio risk figures, e. g. expected losses, unexpected losses and risk contributions. Usually, this information is used for optimising the risk-return profile of the portfolio. We show that gradient-like optimisation methods based on risk contributions can lead to inefficient portfolio structures. To avoid this local optima problem, our optimisation method combines quantitative model features with Genetic Algorithms. The hybrid approach in this paper consists of a task specific Genetic Algorithm that uses special variation operators reflecting portfolio credit risk model knowledge. The method presented here is compatible with any model providing a loss or profit/loss distribution for credit portfolios, e. g. CreditMetrics, CreditRisk+, Wilson's model, and others. As a consequence, it can be used with any risk measure based on such profit-loss distributions like Credit-Value-at-Risk, Expected Shortfall etc. We show how different additional constraints like economic and/or regulatory capital limits can be included in the optimisation process. The results of a test series with different portfolio sizes and structures in a CreditRisk+ General Sector Analysis model framework running on a standard Personal Computer are provided within this paper. They indicate that the hybrid Genetic Algorithm leads to better convergence than a standard Genetic Algorithm approach while not suffering from the local optima problem, and calculates efficient portfolio structures within reasonable time.

Key words: Credit risk, Portfolio credit risk model, Portfolio optimisation, Genetic Algorithm, Credit-Value-at-Risk, Economic capital, Regulatory capital

JEL Classification: G11, G20, G31, G33

[1] **Presenter: Frank Schlottmann**
Institute AIFB
Faculty of Economics
University Karlsruhe
D-76128 Karlsruhe
GERMANY
Tel: +49 721 608 6557
Fax: +49 721 69 37 17
E-Mail: schlottmann@aifb.uni-karlsruhe.de

[2] **Detlef Seese**
Institute AIFB
Faculty of Economics
University Karlsruhe
D-76128 Karlsruhe
GERMANY
Tel: +49 721 608 6037
Fax: +49 721 69 37 17
E-Mail: seese@aifb.uni-karlsruhe.de

Introduction

The problem of portfolio credit risk optimisation is getting more important today because of the intensive development of quantitative portfolio credit risk models since the late 1990s and the increasing trade in financial instruments for transferring credit risk like credit default swaps, asset backed transactions etc. And it will likely be even more important for the financial industry when the new capital adequacy guidelines proposed in the Basle Accord discussion paper [BCBS01] will finally be implemented.
In this paper, we will focus on algorithms for the improvement of the risk-return profile for existing loan portfolios with the restriction of certain capital limits, e. g. imposed by changes of supervisory capital regulations or internal reallocation of risk capital. This kind of loan portfolio management is of great importance especially for, but not limited to, many German and European banks as the typical largest exposures to credit risk for small and medium size universal banks are loans given to companies not having direct access to the capital market. Moreover, there will possibly be more demand for portfolio optimisation as soon as the supervisory capital requirements for banks will be raised by the Basle Committee as it seems to happen according to the current proposal of the new capital adequacy guidelines [BCBS01] e. g. by imposing new 150 percent capital weights on some credit risk exposure types and establishing additional regulatory capital requirements for operational risk.[1] But the methodology presented here is not only applicable under changes of regulatory capital rules. A comparable situation is the reallocation of internal risk capital budgets to different profit centres by the bank's management.
From a complexity researcher's point of view, portfolio credit risk optimisation is an interesting task due to the difficulty of the problem. In the following sections we will develop different algorithms for constrained portfolio credit risk optimisation and compare their performance. We will define a risk adjusted performance measure based on the commonly used Credit-Value-at-Risk for the optimisation of portfolio credit risk including transaction cost. Mainly, we will focus on a framework for the use of heuristic randomised search algorithms called Genetic Algorithms in credit risk contexts that is consistent with quantitative model features. The goal of this framework is to improve convergence compared to a standard Genetic Algorithm approach while avoiding local optima problems.
The paper is organised as follows: At first, we describe our portfolio credit risk optimisation problem and give a short overview of the CreditRisk+ [CSFP97] model features that are applied to the test cases in this paper. Then we give an introduction to Genetic Algorithms as a basis for the description of our Hybrid Genetic-Quantitative Algorithm for solving the optimisation problem. The last section presents some of the empirical results we obtained by running our implementation before a final conclusion and outlook are provided.

1 Portfolio credit risk optimisation problem

1.1 Portfolio credit risk models and overview of CreditRisk+

For risk-return optimisation of loan portfolios, the first question is how to model the dependencies between different obligors. There are different alternatives for modelling portfolio credit risk in general. Among these, CreditMetrics [GFB97], CreditRisk+ [CSFP97] Wilson's model [Wi97a, Wi97b] and the KMV option based approach [Ke98] are intensively discussed in many academic and application-oriented publications. Setting our focus on the default risk of loan portfolios, we will concentrate on CreditRisk+. However, the hybrid

[1] It is not the goal of this paper to criticise the New Capital Accord. Rising supervisory capital requirements are just one scenario for the application of our methodology.

Genetic Algorithm approach developed later is compatible with any other portfolio credit risk model providing a loss (in case of a default mode model) or a profit-loss distribution (in case of a mark-to-market model).
In the following paragraphs, we will give a brief description of the CreditRisk+ General Sector Analysis model for a one year horizon here that concentrates on the main issues concerning our algorithm described later (see [CSFP97, pp. 32-57] for a more detailed derivation of the model). It is an actuarial approach that uses an intensity based modelling of defaults, i. e. the default of each obligor in the portfolio is considered to be a stopping time of a hazard rate process expressed by a Poisson-like process. In case of a default event, the amount of credit exposure (net exposure) lent to the defaulting obligor will be entirely lost. Given are m net exposures of size e_i ($m > 1$, $e_i > 0$, $i, j = 1,..,m$ unless otherwise noted in the remainder of this paper) of different obligors in the portfolio. Each obligor has an associated annual mean default rate p_i (typically, p_i is small: $0 < p_i < 0.1$) and an annual default rate volatility $\sigma_i \geq 0$. Furthermore, there is a total of n independent sectors as common risk factors, where the first sector (k=1) is obligor specific, i. e. in this sector there is no implicit default correlation between obligors ($k = 1,..,n$ unless otherwise noted). The obligors are allocated to the sectors according to sector weights $\Theta_{ik} \in [0,1], \forall i : \sum_{k=1}^{n} \Theta_{ik} = 1$.
The probability generating function (abbreviated PGF) for the losses from the entire portfolio is defined by (cf. [CSFP97, p. 46, formula 61])

$$G(z) := \sum_{i=0}^{\infty} p(\text{aggregated losses} = i \cdot L) \cdot z^i \quad (1.1)$$

where L is a constant defining net exposure bands of constant width [CSFP97, p. 36] and p(...) represents the probability of losing i times the value of L from the whole portfolio.
As the sectors are independent this can be decomposed to (cf. [CSFP97, p. 46, formula 62])

$$G(z) = \prod_{k=1}^{n} G_k(z) \quad (1.2)$$

where $G_k(z)$ is the PGF for the losses from the portfolio in sector k.
To obtain the approximated loss distribution for the portfolio a recurrence relation can be applied to evaluate the coefficients of the PGF (cf. [CSFP97, pp. 45f]). After that, risk figures, e. g. the 99th percentile, can be calculated.
An interesting feature of the model concerning the portfolio optimisation task are the marginal risk contributions [CSFP97, pp. 52-56] of obligor i to the standard deviation of portfolio credit risk:

$$RC_i^{\sigma} := e_i \frac{\partial \sigma_{pf}}{\partial e_i} = \frac{e_i p_i}{\sigma_{pf}} \left(e_i + \sum_{k=1}^{n} \left(\frac{\sigma_k}{\mu_k} \right)^2 e_i p_i \Theta_{ik} \right) \quad (1.3)$$

where σ_{pf} is the portfolio standard deviation derived from the PGF of the portfolio losses, μ_k, σ_k are sector specific parameters calculated directly from the input parameters $e_i, p_i, \sigma_i, \Theta_{ik}$ using definition (1.4) below (note that $\sigma_1 = 0$ by definition of sector 1).

$$\forall k : \mu_k := \sum_{i=1}^{m} \Theta_{ik} p_i, \sigma_1 := 0, \forall k > 1 : \sigma_k := \sum_{i=1}^{m} \Theta_{ik} \sigma_i \quad (1.4)$$

To calculate an approximation for the risk contribution e. g. to the 99th percentile, a scaling factor is defined in the following manner (cf. [CSFP97, p. 53]):

$$\xi_{pf} := \frac{q_{pf}^{0.99} - \mu_{pf}}{\sigma_{pf}} \quad (1.5)$$

where $\mu_{pf}, \sigma_{pf}, q_{pf}^{0.99}$ are the expectation, standard deviation and 99th percentile of the portfolio loss distribution, respectively.
The figures calculated by applying formula (1.3) can be used as a basis for the approximate risk contribution to the 99th percentile by scaling the risk contribution obtained from (1.3) according to ξ_{pf} and adding it to the obligor specific expected loss:[2]

$$RC_i^{0.99} := e_i p_i + \xi_{pf} RC_i^{\sigma} \quad (1.6)$$

We will exploit these figures later when developing our approach to improve calculation speed of our prototype implementation.

1.2 A formal description of the portfolio credit risk optimisation problem

Now we are prepared to specify our portfolio credit risk optimisation problem more formally. Given are the net exposures e_i, their associated p_i and σ_i as well as the sector weights Θ_{ik} for each obligor i. Also, the expected rate of return $r_i \in \mathbb{R}$ and the required regulatory capital percentage $a_i \geq 0$ are known for each net exposure. In case of a transaction of net exposure e_i to a third party, there will be a cost rate (transaction cost, premium etc.) of $t_i > 0$ to be paid by the risk seller. Note that the variables a_i, r_i, t_i are based on the net exposure of obligor i, e. g. $r_i := \frac{\text{absolute expected return from obligor } i}{e_i}$, and not on the total exposure.
Our problem of portfolio credit risk optimisation is to find an optimal portfolio structure according to a chosen risk-return criterion subject to the condition that the supervisory capital requirement for this portfolio structure does not exceed a given supervisory capital limit b_1. In this paper, the following target function scheme was chosen as risk-return criterion:

$$f(\text{portfolio}) := \frac{\text{expected return - expected loss - cost}}{\text{economic capital}} \quad (1.7)$$

The function f is derived from the RAPM coeffient [On99, p. 218] that is commonly used in applications (sometimes called RAROC or RORAC). RAPM stands for Risk Adjusted Performance Measurement and represents the proportion between the expected net return and the risk adjusted capital as a quality measure for portfolio structures.
For risk-return optimisation problems where net exposures can be partially transferred to risk buyers there are different approaches towards solving the corresponding optimisation problem discussed in the literature, e. g. by application of simplex algorithms on a tail conditional expectation risk measure in a simulation model framework (see [AnUr99]) or by using Kuhn-Tucker optimality constraints in a CreditRisk+ model framework (see [Le99]). However, in several real world portfolio optimisation problems the decision is between keeping the obligor in the portfolio or selling the entire exposure to a risk buyer (cf. also [Le99]), or at least the exposure is not arbitrarily dividable into parts. This type of problem is more difficult because in general, integer or binary optimisation problems are typically harder to solve than their relaxation to real numbers. Therefore, we model two-state decision variables for our problem: Either the net exposure remains in the portfolio or the net exposure must be sold entirely to a third party. Consequently, we introduce our decision variables:

$$x_i := \begin{cases} e_i, \text{if } e_i \text{ remains in the portfolio} \\ 0, else \end{cases} \quad (1.8)$$

[2] Note that treating ξ_{pf} as a constant is a slight simplification in this approximation as ξ_{pf} depends itself on e_i but otherwise there is no analytic formula for the risk contribution to a given percentile level according to [CSFP97, p. 53].

By introducing column vectors for the *i*-indexed variables $e_i, p_i, r_i, \sigma_i, t_i, x_i$, i. e. $e = (e_i)_{i=1,\ldots,m}$ etc. as well as a matrix $\Theta = (\Theta_{ik})_{i=1,\ldots,m,k=1,\ldots,n}$, definition (1.7) can be written as:

$$f(x,e,p,\sigma,\Theta,r,t) := \frac{\overbrace{\sum_{i=1}^{m} x_i r_i}^{\text{expected return}} - \overbrace{\sum_{i=1}^{m} x_i p_i}^{\text{expected loss}} - \overbrace{\sum_{i=1}^{m} (e_i - x_i) t_i}^{\text{transaction cost}}}{\underbrace{q_{pf}^{0.99}(x,p,\sigma,\Theta) - \sum_{i=1}^{m} x_i p_i}_{\text{economic capital}}} \quad (1.9)$$

Again, $q_{pf}^{0.99}(x,p,\sigma,\Theta)$ denotes the 99th percentile of the resulting portfolio loss distribution now depending explicitly on the specified input variables, particularly on vector *x*.
The denominator of *f* in (1.9) is known as Credit-Value-at-Risk. It must be mentioned here that there are some weaknesses of Credit-Value-at-Risk concerning additivity of different risk figures, compatibility with basic decision theory, etc. There are other risk measures like tail conditional expectation that are coherent [ADEH99] and have further convenient properties. We chose the Credit-Value-at-Risk because of its popularity in real world applications and because of its non-linearity in general, as well as the possible non-convexity[3] of the search space which makes the problem attractive for Genetic Algorithms and similar problem solving methods that were reported to be successful in other non-linear, non-convex contexts. But we consider our approach to be compatible with other currently known risk measures that provide a risk figure usable as denominator of *f*. As a conclusion from our argument above, the resulting search space induced by the alternative risk measure is particularly not required to be convex.
For a formal definition of the optimisation problem we define a matrix *A* containing the capital weights a_i (based on the net exposures) in the first row and a column vector *b* containing the regulatory capital limit b_1 in its first row:

$$A := (a_{uv}) = \begin{cases} a_v, u = 1 \\ 0, else \end{cases}, u, v = 1, \ldots, m$$
$$b := (b_u) = \begin{cases} b_1, u = 1 \\ 0, else \end{cases}, u = 1, \ldots, m \quad (1.10)$$

Now our portfolio credit risk optimisation problem can be specified as follows:

$$\begin{aligned} &\max f(x,e,p,\sigma,\Theta,r,t) \\ &\text{w.r.t. } Ax \le b \\ &x_i \in \{0, e_i\} \end{aligned} \quad (1.11)$$

This problem is computationally hard as it is equivalent to a binary knapsack problem which is NP-complete, i. e. the problem cannot be solved efficiently in polynomial time with an exact algorithm unless P = NP (see [ScSe01b] for a formal proof and further explanations). Consequently, there is a need for the development of adequate heuristics to solve the problem approximately.

1.3 Risk-return optimisation by gradient based methods

A first canonical approach towards solving the problem is a so-called hill climbing or greedy algorithm that iterates on the gradient of *f* using the risk contributions from formula (1.6) for each obligor. It works as follows:

[3] See e. g. [Pf00] for a discussion of the properties of the Value-at-Risk measure.

Algorithm 1:

Input: $e, p, \sigma, \Theta, r, t, A, b$

Initialisation $\forall i : \widehat{x_i} := e_i$

Do

Copy $\forall i : x_i := \widehat{x_i}$

Calculate $f_{old} := f\left(x, e, p, \sigma, \Theta, r, t\right)$

For all $x_j > 0$ calculate the partial derivatives $d_j := \dfrac{\partial}{\partial x_j} f\left(x, e, p, \sigma, \Theta, r, t\right)$

Choose the minimal gradient component $\hat{i} := \arg\min_j \left\{ d_j \,\middle|\, x_j > 0 \right\}$

Remove the minimal gradient component's exposure from portfolio $\widehat{x_{\hat{i}}} := 0$

Calculate $f_{new} := f\left(\hat{x}, e, p, \sigma, \Theta, r, t\right)$

While $\left(\exists i, j, i \neq j : x_i > 0 \wedge x_j > 0\right) \wedge \left(\left(\sum_{i=1}^{m} x_i a_i > b_1\right) \vee \left(f_{old} < f_{new}\right)\right)$

Output: x

The partial derivative d_j can be obtained by evaluating (see appendix for calculation)[4]

$$d_j := \frac{x_j \left(r_i - p_i + t_i\right)\left(\xi_{pf}\sigma_{pf}\right) - \left(\sum_{i=1}^{m}\left(r_i - p_i + t_i\right)x_i - \sum_{i=1}^{m} e_i t_i\right)\left(\xi_{pf} RC_j^{\sigma}\right)}{x_j \left(\xi_{pf}\sigma_{pf}\right)^2} \quad (1.12)$$

In each iteration of the loop, *Algorithm 1* removes the net exposure with the smallest gradient component from the portfolio. The loop terminates if there are less than two exposures in the portfolio or if the capital restriction given by b_1 is no longer violated and no further improvement of the target function value is made.

For perfectly diversified and/or unconstrained portfolios the results of *Algorithm 1* can be globally optimal. But in general, this gradient based approach can lead to inefficient portfolio structures, i. e. not globally optimal solutions, because the overall dependencies between the obligors' contributions to the target function value and/or the degree of constraint violation cannot be recognised perfectly in each iteration of the loop. In addition to that, *Algorithm 1* may suffer from the fact that it uses approximated risk contributions for partial derivatives as defined by the CreditRisk+ model framework. Note that if *Algorithm 1* provided a globally optimal solution for each instance of problem (1.11), then we could find an algorithm solving an NP-complete problem in polynomial time!

An example for an inefficient portfolio structure resulting from isolated use of gradient based methods (calculated by Credit-Value-at-Risk contributions and return coefficients) for portfolio management is provided in the fourth section of this paper. It shows that even for unconstrained risk-return optimisation the above gradient based approach can produce inefficient solutions.

Nevertheless, *Algorithm 1* is a very fast and straightforward approach that is useful as a benchmark for more complex methods, and it can be combined with other methods to improve their convergence.

In some NP-complete problem contexts, Genetic Algorithms have been applied successfully to find globally near-optimal or optimal solutions within reasonable time. We will give a short

[4] Remember that d_j is calculated for $x_j > 0$. For zero-value x_j variables ($x_j = 0$), x_j must be set to a small positive value to allow calculation of (1.12) if necessary.

introduction to this problem solving methodology in the next section before we develop our algorithm combining genetic and quantitative ideas to solve the portfolio credit risk optimisation problem.

2 Genetic Algorithms

2.1 Introduction to Genetic Algorithms

Genetic Algorithms (GAs) are heuristic randomised search algorithms reflecting the "survival of the fittest" principle that can be observed throughout many evolution processes in nature. This analogy to Darwinian evolution was proposed in the 1960s by Holland (see [Ho75]) in his studies on adaptive systems. Independently, Rechenberg [Re73] and Schwefel [Sc95] investigated evolutionary strategies in combinatorial optimisation. Since then, successful applications of GAs have been reported for many theoretical and real-world problems. Especially their use for solving complex problems, like the NP-hard travelling salesman problem or scheduling problems, successfully attracted the attention of researchers around the world (see [Mi92], [Ni97], [FoMi00], [Hr01]).
GAs work on a set of potential solutions rather than on a single solution. The current set of solutions being processed by a GA at a fixed time step is called population or generation. Each single solution of this population is called individual. To apply a GA to a problem, the decision variables of the problem have to be transformed into genes, i. e. the representation of possible solutions to the problem has to be transformed into a string of numbers or characters. The original representation of a solution is called phenotype, the genetic counterpart is called genotype. For example, a common genetic representation for integer-valued phenotypes are their associated bitstrings in binary notation.
A simple Genetic Algorithm scheme can be described as follows:

Algorithm 2: GA scheme

$t := 0$
Generate initial population $P(t)$
Evaluate $P(t)$
Repeat
    Select individuals from $P(t)$
    Recombine selected individuals
    Mutate recombined individuals
    Generate $P(t+1)$
    $t := t + 1$
    Evaluate $P(t)$
Until termination condition is met
Output: $P(t)$ and/or best individual x^* from all iterations

The initial population $P(0)$ can be generated e. g. by random initialisation of every individual or by an algorithm producing near-optimal solutions. In general, to prevent early convergence to local optima the initial population should be as diverse as possible in the search space of potential solutions.
For evaluation of each genotype in a population the GA requires a quality measure for every possible solution (not necessarily feasible if the problem is constrained) that is usually based on the quality of the corresponding phenotype. This evaluation function (called fitness) is problem specific and therefore has to be designed according to the problem parameters and constraints. During the evolution process the GA selects individuals for reproduction from the current population to its succeeding population according to their fitness value, i. e. the

probability of surviving into the next generation is higher for individuals with higher fitness values compared to other individuals in the same generation. More precisely, there exist a number of different selection schemes for individuals that are based on fitness values, each having its own advantages and disadvantages (see e. g. [FoMi00, pp. 179-181] for details), for instance so-called random proportional selection according to the relative fraction of individual *J*'s fitness of the total fitness sum in the population, or tournament selection, where repeatedly two different individuals are drawn randomly with uniform probability from the population and the one with higher fitness value is selected for reproduction until enough offspring are produced. The intention of selection is to drive the evolution process towards the most promising solutions. Again, the implementation of selection is problem dependent and has to be chosen carefully to avoid problems like premature convergence of populations to local optima if the selection pressure is too high.
Since the GA's task is to explore the search space to find globally optimal solutions the selected individuals from a population are modified using genetic operators, sometimes called variation operators ([FoMi00, p. 173]). A typical variation operator is the one-point crossover, i. e. the gene strings of two selected individuals are cut at a randomly chosen position and the resulting tail parts are exchanged with each other to produce two new offspring.

Example 1:
Consider two integer-valued phenotypes, 5 and 17, that are represented by their binary counterparts of maximum length 5. A crossover operation at random cut point 2 is presented in figure 1.

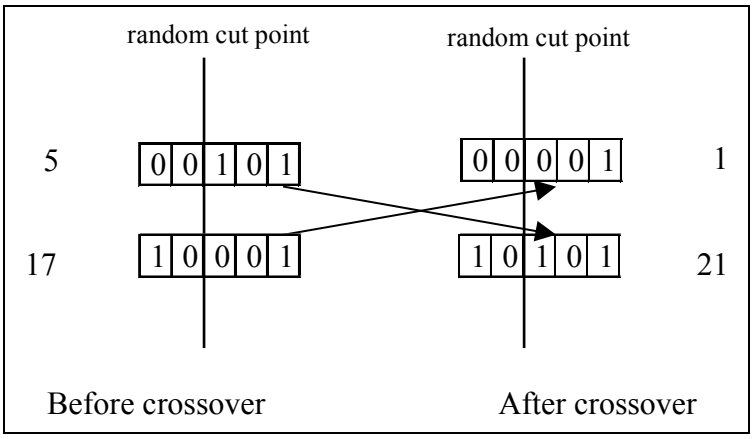

Figure 1: One point crossover

This operation is performed on the individuals selected for reproduction with crossover probability p_{cross}. The main goal of crossover is to move the population through the space of possible solutions.
Moreover, most GAs implement a second genetic operator known as mutation. In analogy to natural mutation, the mutation operator changes the genes of selected individuals randomly with probability p_{mut} (mutation rate) per gene to allow the invention of new, previously undiscovered solutions in the population. Its second task is the prevention of the GA stalling in local optima as there is always a positive probability to leave a local optimum if the mutation rate is greater than zero.

2.2 Advantages of GA applications

As a consequence of using random variation operators and selection schemes, an obvious advantage of GA problem solving is that unlike other methods, it is possible to maintain infeasible and/or suboptimal solutions during the solution process which may be close to the global optimum while there are many local optima far away from the global optimum. In addition to that, GAs are applicable to non-linear, non-convex problems. Further advantages of GAs are (refer to [Fo99] for more details):

- Conceptual simplicity
- Broad applicability
- Outperformance of classical methods on some real problems
- Potential to use problem knowledge and hybridisation
- Parallelism
- Robustness to dynamic changes
- Capability for self-optimisation
- Ability to solve problems without known solutions

Especially the implicit parallelism of GAs could be an advantage using the corresponding technique (see [SBK01]). A theoretical investigation of GAs has not been carried out on a broad basis yet. Nevertheless there are some interesting results e. g. on convergence and modelling by Markov processes (see e.g. [Mu97], [Vo99], [DJW99], [We00], [GrMa00], [EiRu99], [Ru98]).

At this point, we must emphasise that besides the simplicity and the broad applicability of GAs there is no general GA solution that fits to every problem. Like other problem solving methods, the concepts of GAs have to be adapted to the problem to achieve good results, i. e. the genetic representation, the variation operators have to be designed well, etc. Also, the parameters, e. g. mutation rate, crossover probability, must be chosen carefully.

Despite of some research progress that has been made in theoretically justifying the success of GAs in finding good solutions for many problems, some heuristically observed good properties of GAs are still not proved in a strictly mathematical sense. So it is still a matter of empirical work to develop a good GA suitable for a new problem.

2.3 Examples of successful GA applications in finance

GAs have been reported to be successful in different financial problem contexts. Many of these examples inhibit the task of learning or optimising trading strategies for foreign exchange, stock or bond markets (see e.g. [Ba94], [Le95], [FHKNS96], [TsLa00]). Often they are used in conjunction with other methods like Artificial Neural Networks or Fuzzy Logic (see e. g. [HKSZ98]).

Other examples of successful applications are: multi-period asset allocation [BPST00], financial forecasting [Ki97], simulation of stock markets [Ta95], credit evaluation [WHG95], discovery of insider trading [Mo95] and valuation of American put options [Ke00]. For an overview of mean-variance optimisation of stock portfolios using different heuristics including GAs see [CMBS00].

In the next chapter we will introduce our Hybrid Genetic-Quantative Algorithm (written shortly as HGQA) that combines the above Genetic Algorithm methodology with a greedy algorithm based on quantitative information from the portfolio credit risk model similar to *Algorithm 1* from the previous chapter.

3 A Hybrid Genetic-Quantitative Algorithm (HGQA) for constrained portfolio credit risk optimisation

In this section we describe the details of the new HGQA for constrained portfolio credit risk optimisation by subsequently looking at the different parts to be designed before giving a summary of the whole algorithm. The goal of the development of HGQA is to create a well-designed GA consistent with quantitative model features that is combined with an additional quantitative operator to achieve a higher convergence speed of the hybrid system while avoiding convergence to local optima.

3.1 Genotypes and crossover variation operator

For our decision problem given in (1.11), we connect the decision variables x_i to obtain gene strings representing potential solutions. The resulting genotypes consist of real-valued genes connected to strings like the binary strings in figure 1, but instead of taking the values 0 or 1 each gene takes either value 0 or e_i depending on the absence or presence of obligor i in the current solution. So we obtain strings of length m that represent the 2^m combinations of possible (but neither necessarily feasible nor necessarily optimal) portfolio structures.
The number of individuals per population is chosen according to the problem size. For our tests we have set this number to 10 for small test cases and up to 30 for the larger problems.
Since we choose the one point crossover presented in chapter 2 to be the first variation operator for our genotypes, we should keep some quantitative model features in mind to choose a well adapted genetic representation for our phenotypes at this stage. The one point crossover cuts two gene strings at a random position and crosses the tails of the strings to produce two offspring (remember figure 1) with crossover probability p_{cross}. In our test cases we have chosen $p_{cross} = 0.95$ according to empirical observation of results from different p_{cross} settings.
The probability of two genes i, j (these variables represent the index of the genes associated to obligor i and j) from one individual being cut by the crossover increases proportional to the distance $|i-j|$ between the two genes in the gene string as the cut position is determined by a draw from a uniform distribution over m-1 cut possibilities:

$$p(\text{crossover cuts gene } i \text{ and } j) = \frac{1}{m\text{-}1}|i - j| \quad (3.1)$$

For better results of the crossover operator we must ensure that there is a high probability of good partial solutions being recombined with other solutions and not being destroyed by the crossover's cut operation. More formally, we search for a permutation $\pi(i)$ of the portfolio data represented by our genes ensuring a high probability of success for crossover. Therefore we have to remember that the degree of influence between two different obligors i, j (and therefore two different genes) in the CreditRisk+ context particularly depends on sector weights Θ_{ik}, Θ_{jk}, mean default rates p_i, p_j and their volatilities σ_i, σ_j.
More precisely, the pairwise default correlation between two different obligors i, j in the CreditRisk+ model is known to be approximately (cf. [CSFP97, pp. 56-57]):

$$\rho_{ij} \approx \sqrt{p_i p_j} \sum_{k=1}^{n} \Theta_{ik} \Theta_{jk} \left(\frac{\sigma_k}{\mu_k} \right)^2 \quad (3.2)$$

Note that by definition (1.4) for μ_k and σ_k the pairwise default correlation between two obligors i, j also depends on the mean default rates, default rate volatilities and sector weights of obligors other than i, j. But if we consider the proportion $\frac{\sigma_k}{\mu_k}$ to be quite constant during

the optimisation process –this will e. g. be the case if we set $\sigma_i = \omega\mu_i$ for each exposure or more strictly, $\sigma_k = \omega_k\mu_k$ as it is suggested for example in [CSFP97, p. 57] – then differences between distinct pairwise default correlations defined by (3.1) are mainly caused by the product of probabilities p_i, p_j and the product of sector weights Θ_{ik}, Θ_{jk}.

In our target function f (and in the fitness function g) there is no further influence between each pair of two obligors beyond these variables because the expected return, expected loss and transaction cost of each obligor are independent from other obligors' variables.

A good crossover performance under these circumstances is accomplished by arranging the gene order according to the main driving factors of default correlations as derived above. The distance between two genes of obligors sharing higher default correlation should be lower than the distance between genes of two obligors with lower default correlation as the probability of a good partial solution for the highest correlated genes (obligors) being destroyed by the crossover should be kept low. In general, this leads to a problem based on m^2 correlations to be calculated. For large portfolios the calculation of m^2 correlations will take large amounts of computing time. As we want to avoid spending too much time on the correlation calculations, it is reasonable to sort the portfolio data in ascending lexicographic order according to the product of sector weights and mean default rate of each obligor, i. e. the sorting procedure is based on the main drivers of default correlation. To obtain a sorting according to the level of implicit sector correlation, the priority for the ordering relation is set by the quotients $\omega_k := \frac{\sigma_k}{\mu_k}$ (so-called variation coefficient).

This means that the first criterion to be checked in the pairwise comparison between two obligors *i, j* in the sorting procedure is which of the following expressions are true:

$$\left(p_i\Theta_{ik_1} < p_j\Theta_{jk_1}\right) \text{ or } \left(p_i\Theta_{ik_1} > p_j\Theta_{jk_1}\right) \text{ or } \left(p_i\Theta_{ik_1} = p_j\Theta_{jk_1}\right)$$

for the sector $k_1 := \arg\max_k \{\omega_k\}$ with the highest variation coefficient.

If the third expression is true, then the next criterion to be checked for the comparison is the analogous relation between obligor *i*'s and obligor *j*'s product of the mean default rate and the sector weight in the sector k_2 that has the second highest variation coefficient, and so on until all obligors are sorted or all sectors have been processed.

An example sorting operation is shown below:

***Example 2**:*

Given is the following portfolio data:

i	e_i	p_i	Θ_{i1}	Θ_{i2}	Θ_{i3}	$p_i\Theta_{i1}$	$p_i\Theta_{i2}$	$p_i\Theta_{i3}$
1	10000	2%	100%	0%	0%	2%	0%	0%
2	2500	3%	0%	100%	0%	0%	3%	0%
3	30000	2%	0%	0%	100%	0%	0%	2%
4	4300	2%	0%	0%	100%	0%	0%	2%
5	6700	4%	0%	100%	0%	0%	4%	0%
6	7300	4%	0%	0%	100%	0%	0%	4%
7	1100	4%	100%	0%	0%	4%	0%	0%
8	24500	2%	0%	0%	100%	0%	0%	2%
9	3000	2%	100%	0%	0%	2%	0%	0%
10	9400	3%	0%	0%	100%	0%	0%	3%

Table 1: Unsorted portfolio data

Consider $\omega_1 = 0$ by definition of sector $1, \omega_2 = 0.7, \omega_3 = 1$.

Therefore $k_1 = 3, k_2 = 2, k_3 = 1$ is the priority for the sorting procedure.
After running the sorting procedure the portfolio data is ordered as follows:

i	$\pi(i)$	e_i	p_i	Θ_{i1}	Θ_{i2}	Θ_{i3}	$p_i\Theta_{i1}$	$p_i\Theta_{i2}$	$p_i\Theta_{i3}$
1	1	10000	2%	100%	0%	0%	2%	0%	0%
9	2	3000	2%	100%	0%	0%	2%	0%	0%
7	3	1100	4%	100%	0%	0%	4%	0%	0%
2	4	2500	3%	0%	100%	0%	0%	3%	0%
5	5	6700	4%	0%	100%	0%	0%	4%	0%
3	6	30000	2%	0%	0%	100%	0%	0%	2%
4	7	4300	2%	0%	0%	100%	0%	0%	2%
8	8	24500	2%	0%	0%	100%	0%	0%	2%
10	9	9400	3%	0%	0%	100%	0%	0%	3%
6	10	7300	4%	0%	0%	100%	0%	0%	4%

Table 2: Sorted portfolio data

After sorting, the highest correlated obligors are located next to each other within each sector. As the sectors are pairwise independent, now there is a higher probability compared to the unsorted case that a good partial solution found for a sector will be recombined with another good partial solution found for any of the other sectors without destroying the structure of both partial solutions if they are selected for crossover.
This preliminary sorting operation is done on the input data before the core HGQA starts. It is also successful in cases where each obligor is allocated to more than one sector like in our test portfolios described later.
The above theoretical considerations are supported by empirical evidence. In our experiments during development of the HGQA, the performance of the algorithm in terms of convergence was usually[5] better when the input was sorted according to the criterion formulated above. In other words, the HGQA discovered good solutions in earlier generations when the input was prepared in the above mentioned way.
For the application of HGQA in model contexts other than CreditRisk+, the portfolio data should be prepared according to the default correlations in an analogous way. Moreover, our above considerations are also applicable to crossover operators different from the one point variant chosen here.

3.2 Mutation variation operator

The second variation operator in the HGQA is a mutation that changes the state of each gene with mutation rate p_{mut}. If the value of the gene to be mutated is e_i, it will be equal to 0 after mutation and vice versa. In our test cases, we set the mutation rate to $p_{mut} := \frac{2}{m}$. As a consequence, on average two genes of every offspring individual are changed per mutation operation.

3.3 Design of fitness function

For the evaluation of individuals, the fitness function g is derived from the target function f of our optimisation problem (cf. formula (1.9)) by adding a penalty term that reduces the target function if the capital limit is violated:

[5] Remember that a GA is a stochastic algorithm so it is possible (although it is not likely) that even an inferior algorithm will show better performance than its superior counterpart.

$$g(x,e,p,\sigma,\Theta,r,t) := f(x,e,p,\sigma,\Theta,r,t)\cdot\left(1+\operatorname{sign}(f)\min\left\{0;\frac{b_1-\sum_{i=1}^{m}a_i x_i}{b_1}\cdot z\sqrt{1+w}\right\}\right) \quad (3.3)$$

where sign(*f*) is the sign of $f(x,e,p,\sigma,\Theta,r,t)$, z is a constant (in our tests we set $z = 2$) and w is the number of populations since the best solution has become infeasible. w is initialised to 0 at the start of our GA. The first time the best solution gets infeasible w is increased by 1. Afterwards, w is incremented by 1 in each population step until the best solution is feasible and w can be reset to 0.

This modification of the target function *f* penalises infeasible solutions during the search for optimal solutions. Its advantage lies in the fact that it is not necessary to maintain only feasible solutions during the search process - the HGQA can find a feasible optimal solution that is near to an infeasible solution or combine parts of two infeasible near-optimal solutions to a feasible optimal solution. We chose a penalty function proportional to the current result of the RAPM function *f* because the pressure towards feasible solutions should increase during the optimisation process, i. e. for potential solutions with higher values of the target function more weight is put on the supervisory capital constraint. Furthermore, the pressure towards feasible solutions increases approximately proportional to the square root of the number of populations since the best solution has become infeasible.

If there were additional capital constraints to be respected like economic capital limits different from their supervisory counterparts, it would simply require more penalty terms to be added to the fitness function *g*. This would lead to no substantial difference from our work presented herein, so for the sake of clarity, we focus on supervisory capital limits in the remainder of this paper.

3.4 Additional quantitative variation operator

For the additional quantitative component of HGQA, we use a stochastic greedy algorithm as a third variation operator. Similar to *Algorithm 1,* it is based on the first partial derivative (cf. (1.12)) of the function *f*. This greedy algorithm works as follows:

Algorithm 3:

Input: $e, p, \sigma, \Theta, r, t, A, b, P(t)$

For each $x \in P(t)$ apply the following instruction block with probability p_{greedy}

Choose the direction of changes:

If the capital restriction is violated by x then $D := -1$

If the capital restriction is not violated by x then choose between $D := 1$ or $D := -1$ with equal probability 0.5

Initialisation $\forall i : \widehat{x_i} := x_i$

Do

Copy $\forall i : x_i := \widehat{x_i}$

Calculate $f_{old} := f\left(x, e, p, \sigma, \Theta, r, t\right)$

For all x_j calculate the partial derivatives $d_j := \dfrac{\partial}{\partial x_j} f\left(x, e, p, \sigma, \Theta, r, t\right)$

If $D = -1$

Choose the minimal gradient component

$\hat{i} := \arg\min_j \left\{ d_j \,\middle|\, x_j > 0 \right\}$ of exposures currently remaining in the portfolio

Remove this exposure from portfolio: $\widehat{x_{\hat{i}}} := 0$

Else

Choose the maximal gradient component

$\hat{i} := \arg\max_j \left\{ d_j \,\middle|\, x_j = 0 \right\}$ of exposures currently removed from the portfolio

Add this exposure to portfolio: $\widehat{x_{\hat{i}}} := e_{\hat{i}}$

End If

Calculate $f_{new} := f\left(\hat{x}, e, p, \sigma, \Theta, r, t\right)$

While $\left(\exists i, j, i \neq j : x_i > 0 \wedge x_j > 0\right) \wedge$

$$\left(\left(f_{old} < f_{new}\right) \vee \left(D = -1 \wedge \left(\sum_{i=1}^{m} x_i a_i > b_1 \right) \right) \vee \left(D = 1 \wedge \left(\sum_{i=1}^{m} x_i a_i \leq b_1 \right) \right) \right)$$

Replace x in $P(t)$ by its optimised version

End For

Output: $P(t)$

If the current solution x from $P(t)$ to be optimised with probability p_{greedy} is infeasible because the capital restriction is violated, the algorithm will remove the net exposure with the minimum gradient component value from the portfolio. In addition to the penalty term in the fitness function g, this condition ensures that the hybrid search algorithm moves towards feasible solutions. In case of a feasible solution that is to be optimised, the direction of search for a better solution is determined by a draw of a uniformly distributed (0,1)-random variable. This stochastic behaviour helps preventing the algorithm from stalling into the same local optima during the global optimisation process. The algorithm terminates if less than two obligors remain in the portfolio or if the target function value is not improved and the current solution is feasible.

We chose $p_{greedy} = 0.005$ for our test cases. This is a good compromise between improving convergence speed of HGQA and preventing early convergence to local optima. For the

implementation of *Algorithm 3* we use the approximated risk contributions from the CreditRisk+ model as described by formula (1.12) to save computation time for the partial derivatives.
At this point, we should keep in mind that our approach is not restricted to the application of CreditRisk+. If we chose another model for the portfolio calculations the risk contributions that were needed for calculation of the partial derivatives would easily be obtained by a marginal calculation, i. e. the risk contribution for an obligor *i* would be calculated by comparing the two model results for the portfolio with obligor *i* and without obligor *i* in the portfolio. Note that for simulation based approaches like CreditMetrics it is not necessary to repeat the whole simulation for each obligor to get the marginal results. Instead, it is sufficient to subtract the vector of results for obligor *i* over each simulated scenario from the vector of portfolio results over each simulated scenario by components and calculate the risk figures afterwards. This is important concerning calculation speed.

3.5 Selection method

To ensure an adequate pressure towards better solutions in terms of higher fitness values, tournament selection is applied to the parents of each generation to choose the gene strings that are subject to the variation operators. Tournament selection is favourable here because of its lower pressure towards solutions with high fitness values compared to random proportional selection. As the fitness landscape to be discovered by the GA is non-linear and non-convex because of the properties of the Credit-Value-at-Risk in general, exaggerated high pressure towards high fitness solutions probably results in premature convergence of the GA to local optima. Moreover, the greedy component of HGQA puts further pressure towards local convergence on the individuals, so tournament selection is preferable.
These considerations are consistent with empirical observations from some of our test cases where we temporarily applied random proportional selection. Even when deactivating the greedy component there was too much early convergence under the random proportional selection scheme.
After the application of variation operators to *P(t)*, the best strings from parents and offspring are copied to the next generation *P(t*+1*)* (so-called $(\lambda + \mu)$-reproduction).

3.6 Initial population

The initial population of individuals for the optimisation process is generated randomly using uniformly distributed draws from $\{0, e_i\}$ for all genes (obligors) *i* and all individuals to ensure a good a priori distribution over the space of possible solutions because of its potential non-convexity and non-linearity according to the properties of the Credit-Value-at-Risk measure. Furthermore, this avoids premature convergence to local optima having high fitness values at the start of the optimisation process.

3.7 Termination condition

It is important to allow temporary deterioration of fitness values during the evolution of populations since the GA should be able to leave local optima and move to other regions of the search space. However, the evolution of populations must be stopped at a finite time to terminate the GA. We chose a termination condition that stops the evolution of new populations if there has been no improvement of the best individual's fitness during the last $10 \cdot m$ populations, where *m* is still the number of obligors representing an estimate for the size of the problem. This is a compromise between the two above mentioned aspects that was found to be adequate in our tests during development of the implementation.

3.8 Summary of HGQA algorithm

Putting all the elements together, the HGQA reads as follows:

Algorithm 4:

Input: $e, p, \sigma, \Theta, r, t, A, b$

Perform sorting operation to permutate input data according to default correlations
$t := 0$
Generate initial population *P(t)* by random initialisation of individuals
Evaluate *P(t)* using fitness function *g*
Repeat
Select individuals from *P(t)* according to tournament selection for reproduction, mutation and/or greedy algorithm
Recombine selected individuals using one point crossover w. probability p_{cross}
Mutate selected individuals with respect to mutation rate p_{mut} per gene
Apply greedy algorithm with probability p_{greedy} to selected individuals (*Algorithm 3*)
Generate *P(t*+1*)*
$t := t + 1$
Evaluate *P(t)* using fitness function g
Until no improvement of best fitness is made for $10 \cdot m$ generations (*m* is still the dimension of *e*, i. e. the number of obligors)

Output: *P(t)* and/or best individuals from all iterations

In the following section the results obtained by running our prototype implementation of *Algorithm 4* are compared to the results of *Algorithm 1*. In addition, for a small test portfolio, the results are compared to the globally optimal solution calculated by a complete enumeration of the search space.

4 Some Empirical Results

4.1 Implementation details and test scenarios

For evaluation purposes, we have built an implementation of the CreditRisk+ model framework, *Algorithm 4* (including *Algorithm 3*) and *Algorithm 1*, as well as a simple enumeration algorithm (called *Algorithm 5*) that enumerates all possible portfolio structures for our problem (1.11), i.e. the latter algorithm serves as a proof for the globally optimal portfolio structure that should be discovered by the other search algorithms. All tests of the above implementations have been carried out on a standard desktop PC (800 MHz single CPU).

Although more tests cases have been examined during development of the system, we focus on three sample loan portfolios in this paper. The structures of these portfolios were selected in analogy to real world data.[6] Obligors' mean default rates and default rate volatilities are determined by their associated bank-internal rating class. We chose partially unstructured portfolio returns and transaction cost for the portfolios presented below because very regular portfolio structures (i. e. returns, transaction cost, supervisory capital, default rate volatilities perfectly match the default probability structure) were easily solved by our algorithm in other tests. And in real world applications there are many unstructured parameters, as these figures

[6] The detailed structure of portfolio 1 is provided in the appendix. The other test cases can be retrieved via http://www.aifb.uni-karlsruhe.de/CoM/HGQA/tests.html. Note that our r_i variables are calculated by dividing the total return by the net exposure, therefore our returns may seem to be higher than e. g. returns on bonds.

usually result from negotiations between different parties under asymmetric information, often leading to unstructured rates of return and/or transaction cost. For instance, interesting situations inhibit a trade-off between risk and return or a strong capital limit or a combination of these criteria.

Portfolio m20k2: $m = 20$ obligors, $k = 2$ sectors, $b_1 = \frac{\sum_{i=1}^{m} a_i}{1.5}$ regulatory capital limit, i. e. the regulatory capital requirement of the risk-return efficient solution must be $\leq \frac{2}{3}$ of the capital requirement prior to optimisation.

Portfolio m20k2u: same as *Portfolio m20k2* but there is enough regulatory capital so that the capital budget restriction will not be violated by any possible portfolio structure (unrestricted problem).

Portfolio m100k3: $m = 100$ obligors, $k = 3$ sectors, $b_1 = \frac{\sum_{i=1}^{m} a_i}{1.8}$ regulatory capital limit.

Portfolio m500k2: $m = 500$ obligors, $k = 2$ sectors, $b_1 = \frac{\sum_{i=1}^{m} a_i}{2}$ regulatory capital limit.

All constrained test cases inhibit a priori strong supervisory capital restrictions. For instance, *Portfolio m20k2* is a priori a difficult problem for *Algorithm 1* and *Algorithm 4* as there are several infeasible local optima near the feasible risk-return efficient solution having significantly higher target function values than the optimal feasible solution. Remembering that there are 2^m possible portfolio structures that have to be calculated to ensure global optimality of a solution, *Algorithm 5* can only be applied to *Portfolio m20k2* due to limited computation time (e. g. probing 2^{100} combinations will take about 10^{30} seconds if one solution can be calculated in a second). As *Algorithm 4* relies on draws from a pseudo random number generator[7], this algorithm is applied 10 times per portfolio under different seed values.

4.2 Empirical results

For an overview of the results obtained from the application of the different algorithms to our test problems, we use the following notation: x^{**} is a feasible solution providing the best known target function value of f (cf. formula (1.9)) for the portfolio, x^* is the best feasible solution found by the regarded algorithm for the portfolio, t_1 is the time until the best solution x^* was found, t_2 is the time until the algorithm terminated. For HGQA, t_3 denotes the number of populations until x^* was discovered. For all algorithms applied to the respective problem, we calculate the relative error according to the following formula:

$$err := \frac{f(x^{**}) - f(x^*)}{f(x^{**})} \quad (4.1)$$

In addition to that, we calculate the relative improvement of the RAPM target function value $f(x)$ for the best solution x^{**} compared to its value before optimisation.

[7] For the test cases documented here, we have used a combined pseudo random number generator from [LEc88]. A detailed discussion of different pseudo random number generators is provided e. g. in [Ge98].

The results for the unconstrained case *portfolio m20k2u* are shown below (the numbers behind *Algorithm 4* indicate independent runs using different seed values for the pseudo random number generator).

	f(x)*	*err*	t_1	t_2	t_3
Algorithm 1: Greedy	10.39%	0.55%	1s	1s	-
Algorithm 4: HGQA,1	10.45%	0.00%	3s	16s	42
Algorithm 4: HGQA,2	10.45%	0.00%	3s	15s	40
Algorithm 4: HGQA,3	10.45%	0.00%	3s	16s	40
Algorithm 4: HGQA,4	10.45%	0.00%	2s	14s	17
Algorithm 4: HGQA,5	10.45%	0.00%	3s	16s	43
Algorithm 4: HGQA,6	10.45%	0.00%	4s	17s	75
Algorithm 4: HGQA,7	10.45%	0.00%	2s	15s	24
Algorithm 4: HGQA,8	10.45%	0.00%	2s	15s	27
Algorithm 4: HGQA,9	10.45%	0.00%	1s	14s	20
Algorithm 4: HGQA,10	10.45%	0.00%	2s	15s	35
Algorithm 5: Enumeration	10.45%	0.00%	-	2864s	-

Table 3: Results for *portfolio m20k2u*

The global optimum x^{**} = (12700, 15000, 0, 19800, 30100, 30600, 43000, 0, 23500, 9200, 40800, 0, 42100, 27200, 0, 0, 40900, 28000, 0, 0) yielding $f(x^{**}) = 10.45\%$ was discovered by HGQA in each run, but not by the very fast *Algorithm 1*. The latter algorithm found a solution near the global optimum.
As the target function value before optimisation was $f(x) = 8.37\%$ the relative improvement of the RAPM risk-return coefficient by HGQA is 24.85%.
For the constrained problem *portfolio m20k2* the difference between HGQA and *Algorithm 1* is significantly higher as shown in table 4 below.

	f(x)*	*err*	t_1	t_2	t_3
Algorithm 1: Greedy	7.76%	14.41%	1s	1s	-
Algorithm 4: HGQA,1	9.06%	0.00%	3s	15s	57
Algorithm 4: HGQA,2	9.06%	0.00%	2s	13s	34
Algorithm 4: HGQA,3	9.06%	0.00%	4s	15s	59
Algorithm 4: HGQA,4	9.06%	0.00%	2s	13s	25
Algorithm 4: HGQA,5	9.06%	0.00%	3s	15s	33
Algorithm 4: HGQA,6	9.06%	0.00%	3s	15s	40
Algorithm 4: HGQA,7	9.06%	0.00%	3s	15s	57
Algorithm 4: HGQA,8	9.06%	0.00%	4s	16s	65
Algorithm 4: HGQA,9	9.06%	0.00%	7s	17s	117
Algorithm 4: HGQA,10	9.06%	0.00%	8s	19s	100
Algorithm 5: Enumeration	9.06%	0.00%	-	2877s	-

Table 4: Results for *portfolio m20k2*

The global risk-return efficient solution x^{**} = (12700, 15000, 3500, 19800, 30100, 30600, 43000, 0, 0, 9200, 40800, 0, 42100, 27200, 0, 0, 40900, 0, 0, 0) that satisfies the capital restriction was found by HGQA in all independent runs. *Algorithm 1* was obviously leading to an inefficient portfolio structure. This is an interesting result as it shows that particularly for undiversified portfolios containing a couple of large exposures the application of *Algorithm 1* can lead to significant inefficiencies.
As mentioned above, the unconstrained target function value before optimisation was $f(x) = 8.37\%$, so the relative improvement of the RAPM risk-return coefficient by HGQA was 8.24% which is lower compared to the unconstrained test case due to the tight capital constraint.

To get an idea how our algorithm moves through the search space, we present a plot of the fitness function and the capital constraint violation of the best individual for one run of HGQA on *portfolio m20k2* in figure 2.

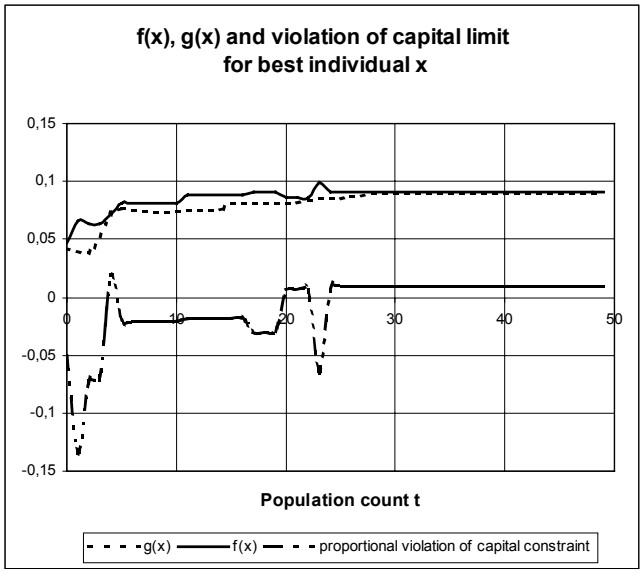

Figure 2: Evolution of target function value $f(x)$, fitness $g(x)$ and violation of capital limit for best individual during 50 populations

The algorithm moved from infeasible solutions to feasible solutions and back to infeasible solutions until it discovered the feasible global optimum. After further population steps without improvement of the best solution, the algorithm terminated. This demonstrates the advantage of temporarily allowing infeasible solutions to evolve in the population.
Table 5 displays the results for *portfolio m100k3*.

	$f(x^*)$	*err*	t_1	t_2	t_3
Algorithm 1: Greedy	15.44%	3.38%	2s	2s	-
Algorithm 4: HGQA,1	15.98%	0.00%	590s	913s	1769
Algorithm 4: HGQA,2	15.98%	0.00%	441s	761s	1286
Algorithm 4: HGQA,3	15.98%	0.00%	94s	413s	257
Algorithm 4: HGQA,4	15.98%	0.00%	214s	537s	633
Algorithm 4: HGQA,5	15.98%	0.00%	364s	681s	1128
Algorithm 4: HGQA,6	15.98%	0.00%	88s	409s	242
Algorithm 4: HGQA,7	15.98%	0.00%	128s	445s	387
Algorithm 4: HGQA,8	15.98%	0.00%	310s	633s	924
Algorithm 4: HGQA,9	15.98%	0.00%	263s	586s	786
Algorithm 4: HGQA,10	15.98%	0.00%	429s	751s	1391

Table 5: Results for *portfolio m100k3*

The best known solution x^{**} provided by HGQA was discovered in all runs despite several infeasible local optima in the neighbourhood of x^{**}. As we chose a very strong capital restriction enforcing the removal of profitable obligors from the portfolio the problem is difficult to solve. This fact is reflected by the very different number of populations that had to be evolved until the best solution was found as the GA part of *Algorithm 4* had to recover from several local optima discovered during evolution. Again, *Algorithm 1* led to an inferior solution due to the difficult problem.
The RAPM coefficient prior to optimisation was $f(x) = 9.49\%$, so the relative improvement achieved by HGQA compared to this value was 40.60%.

Figure 3 shows an example for the difference between the best fitness values obtained by setting $p_{greedy} = 0.005$ (hybrid GA) and by $p_{greedy} = 0.000$ (GA only).

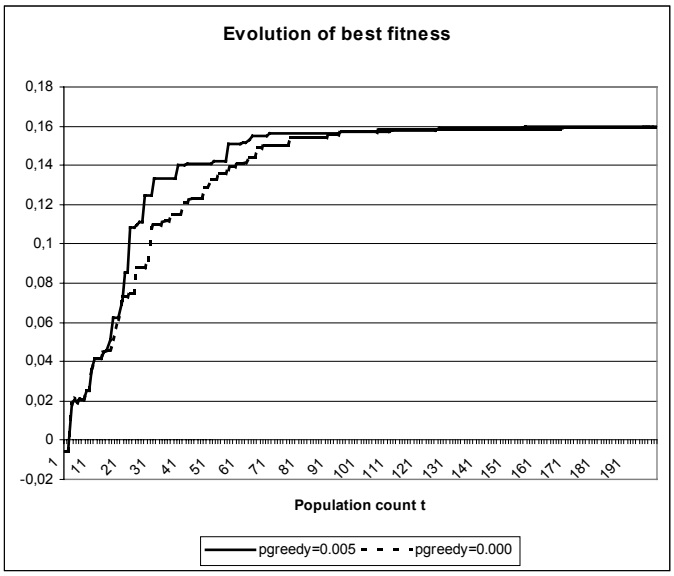

Figure 3: Evolution of fitness function $g(x)$ over 200 populations for $p_{greedy} = 0.005$ and $p_{greedy} = 0.000$

In addition to the obvious difference between the two curves in figure 3 we want to point out that for $p_{greedy} = 0.000$ the GA didn't find the global optimum whereas the HGQA did. This is a typical result as the additional quantitative component of HGQA improves the convergence speed in most cases.
The *portfolio m500k2* test case is a very difficult one as the search space is very large, there is a significant trade-off between risk and return and the a priori tight capital restriction is far away from the best known solution after optimisation, i. e. ex post the best known solution lies deep inside the feasible region surrounded by many near-optimal solutions.
Table 6 shows the results for this problem:

	f(x)*	*err*	t_1	t_2	t_3
Algorithm 1: Greedy	14.31%	4.47%	6s	6s	-
Algorithm 4: HGQA,1	14.95%	0.20%	1.35h	2.15h	6635
Algorithm 4: HGQA,2	14.91%	0.47%	1.92h	3.04h	9773
Algorithm 4: HGQA,3	14.96%	0.13%	1.70h	2.75h	8650
Algorithm 4: HGQA,4	14.94%	0.27%	1.66h	3.87h	8143
Algorithm 4: HGQA,5	14.95%	0.20%	2.13h	2.94h	10542
Algorithm 4: HGQA,6	14.98%	0.00%	1.68h	2.79h	8293
Algorithm 4: HGQA,7	14.98%	0.00%	1.91h	3.39h	9365
Algorithm 4: HGQA,8	14.92%	0.40%	0.61h	1.88h	2873
Algorithm 4: HGQA,9	14.96%	0.13%	1.90h	3.16h	9244
Algorithm 4: HGQA,10	14.94%	0.27%	2.55h	3.70h	12627

Table 6: Results for *portfolio m500k2*

As expected, the again very fast Greedy found a solution significantly away from the best solution x^{**} found by HGQA. Compared to the RAPM target function value of 13.32% prior to optimisation, the latter solution represents a relative improvement of 12.46%.
Due to the complexity of the problem, the HGQA discovered different solutions close to x^{**}. This reminds us that we are using a heuristic algorithm relying on randomised operators. For such large and complex problems it is a non trivial task to find a termination condition for HGQA that ensures reasonable overall runtime of the algorithm on one hand and discovery of

the very best solution in each run on the other hand. We put more weight on the first aspect as our quite strict termination condition served well for most test problems and the above differences between the HGQA results are below an absolute distance of 0.1% and less than a relative error of 0.5%.
Examining the differences between the Greedy algorithm and HGQA we see a rising runtime difference for growing portfolio sizes. It is obvious that the search for better solutions takes more time when the search space size increases exponentially. But we consider a time of 2-3 hours on a single standard PC to be reasonable for a large problem and we have to mention at this point that the best solution for the large portfolio found by HGQA after only a few minutes of running time was even better than the best solution provided by *Algorithm 1*. If one is concerned about a bank typically holding a larger loan portfolio that is to be optimised we have to remember the fact that GAs are well suited for massive parallel implementation, so the runtime will decrease significantly if the problem is being distributed across several CPUs. Another important aspect is that profitable, well-diversified parts of the portfolio can be optimised using the Greedy heuristic, and these parts of the portfolio can be excluded from the HGQA optimisation process as the expected additional improvements compared to the Greedy's results are low. Therefore, one can concentrate on the largest non-profitable and highly correlated obligors where the potential improvements from applying the HGQA are higher.
Finally, we should keep in mind that we are potentially talking about big numbers, so an absolute improvement of 0.5% or 1% expected net risk adjusted return after transaction cost can be a lot of money that is worth spending some computing time!

Conclusion and Outlook

In this paper we have developed different algorithms for constrained portfolio credit risk optimisation and compared their performance. At first, we have created a quantitative heuristic algorithm based on the gradient that we have derived for our chosen RAPM target function. This target function relates net expected return after potential transaction cost for risk transfer to the Credit-Value-at-Risk measure. By reformulating the gradient using risk contributions from the portfolio credit risk model CreditRisk+ we have achieved a high speed implementation. But for difficult problems, this algorithm has led to inefficient portfolio structures as the trade-off between risk, return and the constraints cannot be perfectly solved by such a greedy approach that is based on locally optimal decisions in each optimisation step. Some of our examples have shown that portfolio credit risk optimisation based on risk contributions can lead to inefficient solutions. To prevent this local optima problem we have designed a hybrid system combining quantitative ideas with GA methodology. We have shown a general framework for the application of GAs to portfolio credit risk problems by deriving an appropriate genetic representation consistent with quantitative model features. Furthermore, we have extended this framework to include supervisory capital limits and other constraints.
The tests of our implementation of the hybrid system have indicated higher convergence speed of HGQA than a classic GA approach. Our results documented in this paper have shown that the HGQA outperforms the fast gradient algorithm concerning the quality of the RAPM target function value. In all tests the application of HGQA led to significant improvements of the risk-return target function. For a small test case, we have proved global optimality of HGQA's best constrained and unconstrained solution by an enumeration of the search space. The test cases imply that this algorithm solves difficult problems containing tight constraints and many local optima. Although the implementation has been running on a

single standard desktop PC, the algorithm has found efficient or near-optimal solutions within reasonable time.
Remembering the fact that GAs are well suited for parallel implementation there are good perspectives for improving the speed of future implementations of our framework by using more than one CPU for the evolution of populations. Further research from the viewpoint of risk modelling can e. g. extend the framework presented here by exploiting the latest developments in the CreditRisk+ context published in [BKW01] to include severity variations concerning the net exposures or use an alternative way of calculating the loss percentiles as proposed by [Go01]. Of course, the system can be extended to other exposure types, e. g. in a mark-to-market model context. Due to the flexibility of the GA, many further constraints of practical interest can be easily integrated into our framework, e. g. the simultaneous use of different economic capital limits per rating category and/or industry in the optimisation process. Even more sophisticated restrictions can be handled, e. g. a minimum overall quality of the parts of the portfolio to be sold in an Assed Backed Security transaction which is itself calculated using a non-linear risk model during each optimisation step.
Finally, the system presented in this paper can be integrated into a larger decision support system for risk-return optimisation in a financial institution that supports human portfolio risk-return managers and traders using agent technology as proposed in [ScSe01a].

Acknowledgement

The authors like to thank Michael Lesko and Stephan Vorgrimler for fruitful discussions and reading the manuscript. We also thank Tobias Dietrich and Thomas Stuempert for final corrections. Furthermore, support from GILLARDON financial software GmbH is hereby gratefully acknowledged.

Appendix

A.1 Proof of formula (1.12)

Given is a portfolio specified by the tupel $(x, e, p, \sigma, \Theta, r, t)$.

The risk-return function *f* can be transformed as follows:

$$f(x,e,p,\sigma,\Theta,r,t) := \frac{\sum_{i=1}^{m} x_i r_i - \sum_{i=1}^{m} x_i p_i - \sum_{i=1}^{m} (e_i - x_i) t_i}{q_{pf}^{0.99}(x,p,\sigma,\Theta) - \sum_{i=1}^{m} x_i p_i} \quad (5.1)$$

$$= \frac{\sum_{i=1}^{m} x_i (r_i - p_i) - \left(\sum_{i=1}^{m} e_i t_i - \sum_{i=1}^{m} x_i t_i \right)}{q_{pf}^{0.99}(x,p,\sigma,\Theta) - \sum_{i=1}^{m} x_i p_i} \quad (5.2)$$

$$= \frac{\sum_{i=1}^{m} x_i (r_i - p_i) + \sum_{i=1}^{m} x_i t_i - \sum_{i=1}^{m} e_i t_i}{q_{pf}^{0.99}(x,p,\sigma,\Theta) - \sum_{i=1}^{m} x_i p_i} \quad (5.3)$$

$$= \frac{\sum_{i=1}^{m} x_i (r_i - p_i + t_i) - \sum_{i=1}^{m} e_i t_i}{q_{pf}^{0.99}(x,p,\sigma,\Theta) - \sum_{i=1}^{m} x_i p_i} \quad (5.4)$$

If we calculate a constant multiplier for the given portfolio

$$\xi_{pf} := \frac{q_{pf}^{0.99}(x,p,\sigma,\Theta) - \mu_{pf}(x,p,\sigma,\Theta)}{\sigma_{pf}(x,p,\sigma,\Theta)} \quad (5.5)$$

which can be abbreviated by

$$\xi_{pf} := \frac{q_{pf}^{0.99} - \mu_{pf}}{\sigma_{pf}} \quad (5.6)$$

in analogy to [CSFP97, p. 63], the 99th percentile function can be reformulated by

$$q_{pf}^{0.99} = \mu_{pf} + \xi_{pf} \sigma_{pf} \quad (5.7)$$

By substituting the 99th percentile function in formula (5.4) according to (5.7) we obtain:

$$\frac{\sum_{i=1}^{m} x_i (r_i - p_i + t_i) - \sum_{i=1}^{m} e_i t_i}{\mu_{pf} + \xi_{pf} \sigma_{pf} - \sum_{i=1}^{m} x_i p_i} \quad (5.8)$$

Taking into account that

$$\mu_{pf} := \sum_{i=!}^{m} x_i p_i \quad (5.9)$$

formula (5.8) can be simplified to

$$\frac{\sum_{i=1}^{m} x_i (r_i - p_i + t_i) - \sum_{i=1}^{m} e_i t_i}{\xi_{pf} \sigma_{pf}} \quad (5.10)$$

The partial derivative of *f* is calculated by deriving (5.10) using quotient rule:

$$d_j := \frac{\partial}{\partial x_j} f\left(x, e, p, \sigma, \Theta, r, t\right)$$

$$= \frac{\left(r_j - p_j + t_j\right)\left(\xi_{pf}\sigma_{pf}\right) - \left(\sum_{i=1}^{m}\left(r_i - p_i + t_i\right)x_i - \sum_{i=1}^{m} e_i t_i\right)\left(\frac{\partial}{\partial x_j}\left(\xi_{pf}\sigma_{pf}\right)\right)}{\left(\xi_{pf}\sigma_{pf}\right)^2} \quad (5.11)$$

For $x_j \neq 0$ formula (5.11) is equivalent to

$$\frac{x_j\left(r_j - p_j + t_j\right)\left(\xi_{pf}\sigma_{pf}\right) - x_j\left(\sum_{i=1}^{m}\left(r_i - p_i + t_i\right)x_i - \sum_{i=1}^{m} e_i t_i\right)\left(\frac{\partial}{\partial x_j}\left(\xi_{pf}\sigma_{pf}\right)\right)}{x_j\left(\xi_{pf}\sigma_{pf}\right)^2}$$

$$= \frac{x_j\left(r_j - p_j + t_j\right)\left(\xi_{pf}\sigma_{pf}\right) - \left(\sum_{i=1}^{m}\left(r_i - p_i + t_i\right)x_i - \sum_{i=1}^{m} e_i t_i\right)\left(x_j\frac{\partial}{\partial x_j}\left(\xi_{pf}\sigma_{pf}\right)\right)}{x_j\left(\xi_{pf}\sigma_{pf}\right)^2} \quad (5.12)$$

Recalling the assumption that ξ_{pf} is considered a constant calculated from the portfolio mean and standard deviation, the previous formula can be transformed into

$$\frac{x_j\left(r_j - p_j + t_j\right)\left(\xi_{pf}\sigma_{pf}\right) - \left(\sum_{i=1}^{m}\left(r_i - p_i + t_i\right)x_i - \sum_{i=1}^{m} e_i t_i\right)\left(\xi_{pf}x_j\frac{\partial}{\partial x_j}\sigma_{pf}\right)}{x_j\left(\xi_{pf}\sigma_{pf}\right)^2} \quad (5.13)$$

Finally, remembering that

$$RC_j^{\sigma} := x_j\frac{\partial\sigma_{pf}}{\partial x_j}$$

the substitution of the partial derivative yields to

$$d_j = \frac{x_j\left(r_j - p_j + t_j\right)\left(\xi_{pf}\sigma_{pf}\right) - \left(\sum_{i=1}^{m}\left(r_i - p_i + t_i\right)x_i - \sum_{i=1}^{m} e_i t_i\right)\left(\xi_{pf}RC_j^{\sigma}\right)}{x_j\left(\xi_{pf}\sigma_{pf}\right)^2} \quad (5.14)$$

A.2 Specification of *portfolio m20k2*

i	e_i	Θ_{i1}	Θ_{i2}	p_i	σ_i	r_i	t_i	a_i
1	12700	89%	11%	2.0%	1.0%	4.72%	1.82%	12.60%
2	15000	73%	27%	2.0%	1.0%	3.33%	1.86%	10.00%
3	3500	71%	29%	4.0%	2.0%	2.86%	0.94%	8.57%
4	19800	54%	46%	3.0%	1.5%	5.05%	1.36%	12.63%
5	30100	29%	71%	2.0%	1.0%	8.31%	1.32%	12.96%
6	30600	75%	25%	6.0%	3.0%	7.52%	1.69%	12.09%
7	43000	37%	63%	3.0%	1.5%	4.19%	1.85%	9.30%
8	22800	68%	32%	6.0%	3.0%	7.02%	1.03%	14.04%
9	23500	53%	47%	5.0%	2.5%	5.11%	1.69%	8.51%
10	9200	39%	61%	4.0%	2.0%	14.13%	1.62%	15.22%
11	40800	32%	68%	4.0%	2.0%	6.13%	1.73%	9.07%
12	26200	58%	42%	7.0%	3.5%	4.20%	1.09%	10.69%
13	42100	24%	76%	4.0%	2.0%	5.46%	1.61%	8.79%
14	27200	39%	61%	5.0%	2.5%	7.72%	0.73%	11.40%
15	1900	44%	56%	6.0%	3.0%	5.26%	2.00%	10.53%
16	34700	27%	73%	5.0%	2.5%	4.03%	1.05%	8.65%
17	40900	22%	78%	5.0%	2.5%	8.80%	1.90%	9.29%
18	28000	14%	86%	5.0%	2.5%	6.43%	0.61%	8.93%
19	32200	8%	92%	5.0%	2.5%	2.80%	1.50%	8.70%
20	4800	7%	93%	5.0%	2.5%	4.17%	1.04%	8.33%

Table 7: Data for *portfolio m20k2*

Note that the figures a_i, r_i, t_i are calculated by dividing absolute values by the net exposure e_i. The portfolio data is already sorted according to the criterion derived in section (3.1).

References

[ADEH99] Artzner, P., Delbaen, F., Eber. J., Heath, D.: Coherent measures of risk, in: Mathematical Finance 9 (1999) 3, pp. 203-228.

[AnUr99] Andersson, F., Uryasev, S.: Credit risk optimization with conditional value-at-risk criterion, working paper, Gainesville: University of Florida, 1999.

[Ba94] Bauer, R.: Genetic algorithms and investment strategies, New York: John Wiley & Sons, 1994.

[BCBS01] Basle Committee for Banking Supervision: The new Basel Capital Accord, Consultative Document, Basle, January 2001, http://www.bis.org/publ/bcbsca.htm.

[BPST00] Baglioni, S., da Costa Pereira, C., Sorbello, D., Tettamanzi, A.: An evolutionary approach to multiperiod asset allocation, in: Poli, R., Banzhaf, W., Langdon, W., Miller, J., Nordin, P., Fogarty, T. (ed.): Genetic Programming, Proceedings of EuroGP 2000, Heidelberg: Springer, 2000, pp. 225-236.

[BKW01] Buergisser, P., Kurth, A., Wagner, A.: Incorporating severity variations into credit risk, in: Journal of Risk 3 (2001) 4, pp. 5-31.

[CMBS00] Chang, T., Meade, N., Beasley, J., Sharaiha, Y.: Heuristics for cardinality constrained portfolio optimisation, in: Computers & Operations Research 27 (2000), pp. 1271-1302.

[CSFP97] CreditSuisse Financial Products: CreditRisk+^{TM} - a credit risk management framework, 1997, http://www.csfp.co.uk/creditrisk/assets/creditrisk.pdf.

[Da99] Dawid, H.: Adaptive learning by Genetic Algorithms, second edition, Heidelberg: Springer, 1999.

[DJW99] Droste, S., Janses, T., Wegener, I.: Perhaps not a free lunch but at least a free appetiser, in: Banzaf, W., Daida, J., Eiben, A., Garzon, M., Honarir, V., Jakiela, M., Smith, R. (eds.): Proc. 1st Genetic and Evolutionary Computation Conference, San Francisco: Morgan Kaufmann, 1999, pp. 833-839.
[EiRu99] Eiben, A., Rudolph, G.: Theory of evolutionary algorithms: a bird's eye view, in: Theoretical Computer Science (1999) 229, pp. 3-9.
[FHKNS96] Frick, A., Herrmann, R., Kreidler, M., Narr, A., Seese, D.: A genetic based approach for the derivation of trading strategies on the German stock market, in: Proceedings ICONIP '96, Heidelberg: Springer, 1996, pp. 766-770.
[Fo99] Fogel, D.: An introduction to evolutionary computation and some applications, in: Miettinen, K., Neittaanmäki, P., Mäkelä, M., Periaux, J. (eds.): Evolutionary algorithms in engineering and computer science, Chichester: John Wiley & Sons, 1999, pp. 23-41.
[FoMi00] Fogel, D., Michalewicz, Z.: How to solve it – modern heuristics, Heidelberg: Springer, 2000.
[Ge98] Gentle, J.: Random number generation and Monte Carlo methods, New York: Springer, 1998.
[GFB97] Gupton, G.; Finger, C.; Bhatia, M.: CreditMetrics™ Technical Document, http://www.riskmetrics.com.
[Go01] Gordy, M.: Calculation of higher moments in CreditRisk+ with applications, Washington: Federal Reserve Board, working paper, 2001, http://mgordy.tripod.com.
[GrMa00] Greenhalgh, D., Marshall, S.: Convergence criteria for genetic algorithms, in: SIAM J. Computing 30 (2000) 1, pp. 269-282.
[HKSZ98] Herrmann, R., Kreidler, M., Seese, D., Zabel. K.: A fuzzy-hybrid approach to stock trading, in: Proceedings ICONIP '98, IOS Press, 1998, pp. 1028-1032.
[Ho75] Holland, J.: Adaptation in natural and artificial systems, Ann Arbor: Michigan University Press, 1975.
[Hr01] Hromkovic, J.: Algorithmics for hard problems, Heidelberg: Springer, 2001.
[Ke98] Kealhofer, S.: Portfolio management of default risk, San Francisco: KMV Corporation, 1998.
[Ke00] Keber, C.: Option valuation with the Genetic Programming approach, in: Abu-Mostafa, Y., LeBaron, B., Lo, A., Weigend, A.: Computational finance 1999, Cambridge: MIT Press, 2000, pp. 690-703.
[Ki97] Kingdon, J.: Intelligent systems and financial forecasting, Heidelberg: Springer, 1997.
[LEc88] L'Ecuyer, P.: Efficient and portable combined random number generators, in: Communications of the ACM 31 (1988), pp. 742-749.
[Le95] Levitt, M.: Machine learning for foreign exchange trading, in: Refenes, A. (ed.): Neural Networks in the capital markets, Chichester: John Wiley & Sons, 1995, pp. 233-243.
[Le99] Lehrbass, F.: Rethinking risk-adjusted returns, in: Credit Risk Special Report, Risk (1999) 4, pp. 35-40.
[Mi92] Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs, Heidelberg: Springer, 1992.
[Mo95] Mott, S.: Insider dealing detection at the Toronto Stock Exchange Modelling artificial stock markets using genetic algorithms, in: Goonatilake, S., Treleaven, P. (eds.): Intelligent systems for finance and business, New York: John Wiley & Sons, 1995, pp. 135- 144.
[Mu97] Muehlenberg, H.: Genetic Algorithms, in: Aarts, E., Lenstra, J. (eds.): Local search in combinatorial optimization, New York: John Wiley & Sons, 1997, pp. 137-171.
[Ni97] Nissen, V.: Einführung in Evolutionäre Algorithmen, Braunschweig: Vieweg, 1997 (in German).
[On99] Ong, M.: Internal credit risk models, London: Risk Books, 1999.

[Pf00] Pflug, G.: Some remarks on the Value-at-Risk and the Conditional Value-at-Risk, in: Uryasev, S. (ed.): Probabilistic constrained optimization, Dordrecht: Kluwer, 2000, pp. 272-281.
[Re73] Rechenberg, I.: Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Information, Freiburg: Fromman, 1973 (in German).
[Ru98] Rudolph, G.: Finite Markov chain results in evolutionary computation: A tour d'horizon, in: Fundamenta Informaticae (1998), pp. 1-22.
[SBK01] Schmeck, H., Branke, J., Kohlmorgen, U.: Parallel implementations of evolutionary algorithms, in: Zomaya, A., Ercal, F., Olariu, S. (eds.): Solutions to parallel and distributed computing problems, New York: John Wiley & Sons, 2001, pp. 47-68.
[Sc95] Schwefel, H.: Evolution and Optimum Seeking, Chichester: John Wiley & Sons, 1995.
[ScSe01a] Schlottmann, F., Seese. D.: An agent-based architecture for the management and trading of portfolio credit risk, working paper, Universitaet Karlsruhe, 2001 (in German).
[ScSe01b] Schlottmann, F., Seese. D.: Some decisions in risk management and NP-completeness: Examples and consequences, working paper, Universitaet Karlsruhe, 2001.
[Ta95] Tayler, P.: Modelling artificial stock markets using genetic algorithms, in: Goonatilake, S., Treleaven, P. (eds.): Intelligent systems for finance and business, New York: John Wiley & Sons, 1995, pp. 271-287.
[TsLa00] Tsang, R., Lajbcygier, P.: Optimisation of technical trading strategy using split search Genetic Algorithms, in: Abu-Mostafa, Y., LeBaron, B., Lo, A., Weigend, A. (eds.): Computational finance 1999, Cambridge: MIT Press, 2000, pp. 690-703.
[Vo99] Vose, M.: The simple Genetic Algorithm, Cambridge: MIT Press, 1999.
[We00] Wegener, I.: On the expected runtime and the success probability of evolutionary algorithms, in: Proc. 26th WG 2000, Lecture Notes in Computer Science 1928, Heidelberg: Springer, 2000, pp. 1-10.
[WHG95] Walker, R., Haasdijk, E., Gerrets, M.: Credit evaluation using a genetic algorithm; in: Goonatilake, S., Treleaven, P. (eds.): Intelligent systems for finance and business, New York: John Wiley & Sons, 1995, pp. 39-59.
[Wi97a] Wilson, T.: Portfolio Credit Risk (I), in: Risk (1997) 9, pp. 111-119.
[Wi97b] Wilson, T.: Portfolio Credit Risk (II), in: Risk (1997) 10, pp. 56-61.