CENTRO STUDI BANCA E FINANZA

www.cefin.unimore.it

CEFIN Working Papers

No 15

Optimization Heuristics for Determining Internal Rating Grading Scales

by M. Lyra, J. Paha, S. Paterlini, P. Winker

March 2009

CEFIN – Centro Studi di Banca e Finanza
Dipartimento di Economia Aziendale – Università di Modena e Reggio Emilia
Viale Jacopo Berengario 51, 41100 MODENA (Italy)
tel. 39-059.2056711 (Centralino) fax 059 205 6927

Optimization Heuristics for Determining Internal Rating Grading Scales

Marianna Lyra* Johannes Paha† Sandra Paterlini†
Peter Winker†

Abstract

Basel II imposes regulatory capital on banks related to the default risk of their credit portfolio. Banks using an internal rating approach compute the regulatory capital from pooled probabilities of default. These pooled probabilities can be calculated by clustering credit borrowers into different buckets and computing the mean *PD* for each bucket. The clustering problem can become very complex when Basel II regulations and real-world constraints are taken into account. Search heuristics have already proven remarkable performance in tackling this problem. A Threshold Accepting algorithm is proposed, which exploits the inherent discrete nature of the clustering problem. This algorithm is found to outperform alternative methodologies already proposed in the literature, such as standard k-means and Differential Evolution. Besides considering several clustering objectives for a given number of buckets, we extend the analysis further by introducing new methods to determine the optimal number of buckets in which to cluster banks' clients.

Keywords: credit risk, probability of default, clustering, Threshold Accepting, Differential Evolution.

*Department of Economics, University of Giessen, Germany.
†Dept. of Economics, CEFIN and RECent, University of Modena and Reggio R., Italy.

1 Introduzione Non Tecnica

Il secondo Accordo di Basilea richiede alle banche di accantonare capitale regolamentare quale forma di copertura verso possibili perdite derivanti da un inaspettato elevato numero di default (insolvenza). L'approccio interno per la determinazione del capitale regolamentare richiede in primis di quantificare quale sia la probabilitá di default di ogni singolo cliente ovvero la probabilitá che il singolo cliente sia insolvente nei successivi 12 mesi. Quindi, i singoli clienti devono essere accorpati in "buckets" o classi e ad essi viene assegnati la medesima probabilitá di default "pooled". La determinazione del raggruppamento dei clienti in differenti classi puó essere interpretato statisticamente come un problema di clustering: i clienti sono raggruppati in modo tale da cercare di avere classi omogenee al loro interno ed eterogenee fra loro. In questo lavoro, estendiamo due lavori precedenti di Krink, Paterlini e Resti (2007,2008) raffinando la metodologia utilizzata per la determinazione delle classi e introducendo due nuovi metodi per la determinazione del numero ottimale di clusters o buckets in cui partizionare i clienti.
Interpretando il problema di raggruppamento dei clienti in classi omogenee quale problema di ottimizzazione, emerge l'esigenza di determinare quale metodo utilizzare per la determinazione della soluzione ottima. In questo contesto, il problema di ottimizzazione e' complicato anche dalla presenza di vincoli istituzionali (e.g. la necessitá di avere una probabilitá di default "pooled" di almeno tre punti base) e di vincoli imposti al fine di ottenere ragionevali raggruppamenti (e.g. avere un numero sufficiente di clienti in ogni classe al fine di validazione del modello). Le euristiche di ricerca costituiscono un possibile metodo per la risoluzione di tale problema di ottimizzazione. Tali euristiche si fondano sull'idea di iniziare la ricerca da una possibile soluzione o piú soluzioni all'interno dello spazio di ricerca e di raffinare in modo iterativo tale soluzione fintanto che un dato criterio di convergenza sia soddisfatto. Numerose euristiche di ricerca sono state proposte nella letteratura scientifica e data la recente disponibilitá di sempre piú efficienti risorse computazionali il campo di ricerca é in continua evoluzione. In questo lavoro, mostriamo come l'utilizzo dell'algoritmo di Threshold Accepting sia particolarmente indicato per questo tipo di problema di ottimizzazione di natura discreta e possa essere una alternativa piú rapida e robusta a precedenti metodi proposti in letteratura.

2 Introduction

The second Basel Accord on Banking Supervision requires banks to hold a minimum level of shareholders' capital in excess of provisions. This regulatory capital (RC) may be regarded as some form of self-insurance (in excess of provisions) against the consequences of an unexpectedly high number of defaults. This amount of capital depends on the exposure to risk of the bank. Financial intermediaries have to assess the clients' riskiness by evaluating their probability of default (PD), i.e., the probability that a borrower will default over the subsequent 12 months. Afterwards, clients are pooled together in buckets (PD-buckets) and are assigned the same "pooled" PD. While many studies have been devoted to the phases of rating assignment, quantification, and validation, the problem of determining the width and the number of PD buckets has received much less attention. We propose to fill this gap in the literature by proposing an error-based statistical methodology to determine the optimal structure of PD-buckets. Thereby, we consider the problem of determining the PD-buckets as a clustering problem, where the aim is to find the cluster structure that allows to minimize a given error measure under the relevant real-world constraints. Previous related work can be found, e.g., in Foglia et al. (2001), Paterlini and Krink (2006), Krink et al. (2007) and Krink et al. (2008). We extend the analysis mainly in two directions.

First, we propose a methodology not only to tackle the problem of determining the PD buckets width, but also to determine the optimal number of buckets in which to partition the banks' clients. This problem is complex to tackle since there is a trade-off between having a small number of large buckets and a high number of small buckets. In fact, clients belonging to the same buckets are assigned the same pooled PD. Hence, we would like to have a large number of buckets in order to minimize the loss of precision. However, in such a case it would be difficult to validate the consistency of the rating scheme ex post, since the number of defaults in each bucket would probably be too low for statistical validation. On the contrary, if the number in which to partition the clients is small, buckets tend to be too wide which might lead to an overstatement of the capital charge, given the concave shape of the capital function (Kiefer and Larson, 2004), and to opportunistic behavior and adverse selection of clients.

Second, we introduce the Threshold Accepting (TA) algorithm (Dueck and Scheuer, 1990; Winker, 2001; Gilli and Winker, 2004) in order to de-

termine the optimal *PD* buckets structure. Compared to the Differential Evolution (DE) heuristic, which has been shown in a previous study to have superior performance with respect to k-means, Genetic Algorithms, Particle Swarm Optimization and Random Search (Krink et al., 2007), TA is particularly suited for discrete search spaces. By exploiting the discreteness of the search space of the *PD* bucketing problem, it avoids to search on plateaus of the objective function, but can still deal with local minima. Our extensive investigation on a real-world dataset shows that TA can be a faster and more robust alternative to DE.

The paper is structured as follows. Section 3 introduces the formal framework for the error-based approach to PD bucketing by considering the regulations put forward by the Basel II accord and some other real-world constraints. Several objective functions and constraints for the optimization problem are presented. Section 4 describes Threshold Accepting. Empirical results and performance comparison are then reported in Section 5. Section 6 extends our formal framework by introducing the endogenous choice of the optimum number of buckets and discusses some results. Finally, Section 7 concludes and suggests further research perspectives.

3 Basel II and Clustering of Credit Risk

The framework of the second Basel Capital Accord (Basel II) puts a strong emphasis on the adequacy of banks' equity for a given risk profile. According to Basel II (Basel Committee on Banking Supervision, 2005) a bank's potential loss if all borrowers default is the sum of these borrowers' exposure at default (EAD) times the fraction (loss given default - LGD) of EAD that may not be recovered. Basel II requires banks to hold sufficient capital in order to cope with losses of a certain size such that a bank's probability of going insolvent is driven below some confidence level.The bank's value at risk (VaR) is defined as its potential loss, excluding values above the c-th percentile, where c is the confidence level. A bank may account for the expected loss (i.e., $EAD \times LGD \times PD$) by provisioning. However, under sufficiently negative economic conditions the conditional (also called stressed or downturn) probability of default (PD_c) is likely to exceed PD and thus may cause losses in excess of provisions. In order to ensure the stability of the banking system, banks are required by Basel II to hold regulatory capital (RC) that is related to these unexpected losses. For determining RC

borrowers have to be assigned to at least seven internal borrower grades b (also called groups or buckets) for non-defaulted borrowers based on their creditworthiness. Then, RC can be computed by, e.g., treating the mean PD ($\overline{PD}_b$) of all borrowers in bucket b as a proxy of an individual borrower's PD. We assume that a bank employs a statistical default prediction model so that an estimate for each borrower's individual PD is available. Then, RC for an individual borrower $RC(PD_i)$, when no maturity adjustment is considered, is given by equation (1) where the stressed PD ($PD_{c,i}$) is given by equation (2). Φ and Φ^{-1} denote the cumulative standard normal density function and its inverse, respectively. The asset correlation R reflects how the individual PDs are linked together by the general state of the economy, the firm's size (as measured by sales) and the size of their EAD.

$$RC(PD_i) = 1.06 \cdot EAD_i \cdot LGD \cdot (PD_{c,i} - PD_i)\,, \tag{1}$$

$$PD_{c,i} = \Phi\left(\frac{\Phi^{-1}(PD_i) - \sqrt{R_i} \cdot \Phi^{-1}(0.001)}{\sqrt{1 - R_i}}\right). \tag{2}$$

If a borrower i is assigned to bucket b her conditional PD ($PD_{c,i,b}$) can be determined by replacing PD_i with $\overline{PD}_b$ in equation (2). The sum of RC for all borrowers i over all buckets b may be computed as (3):

$$RC = \sum_b \sum_i 1.06 \cdot EAD_i \cdot LGD \cdot \left(PD_{c,i,b} - \overline{PD}_b\right). \tag{3}$$

In our implementation we compute the asset correlation according to paragraph 273 of the Basel II framework by normalizing debtors' sales to EUR 5 million if they are below that threshold and to EUR 50 million if they are above this threshold. Consequently, we do not treat small firms' exposures as retail exposures as stated in paragraph 232. 1.06 is an empirically derived scaling factor that prevents RC calculated under Basel II to drop below RC under the Basel I framework. Computing RC from pooled PDs as shown above results in an approximation error. Therefore, Basel II requires banks to perform credit risk rating, i.e., assigning borrowers to buckets *meaningfully*. On the one hand, this means to maximize the homogeneity of borrowers within a given bucket. This may be done by grouping borrowers in minimizing some objective function using an optimization technique as described in Section 4. On the other hand, adjacent buckets must be clearly distinguishable, i.e., heterogeneous. There is a trade-off between

homogeneity and heterogeneity since increasing the number of buckets is likely to decrease heterogeneity within buckets but raise homogeneity between buckets. This trade-off as well as the necessity to ex post validate the meaningfulness of the credit risk rating system leads us to the question which number of buckets to choose. We will address this issue in Section 6.

The goal of maximizing within-buckets homogeneity may be operationalized by different objective functions. First, one may minimize the squared error that arises from substituting a borrower's individual *PD* by the mean of its bucket. This may be done using unconditional *PD*s (point-in-time approach) resulting in the following objective function:

$$min \sum_{b} \sum_{i \in b} \left(PD_i - \overline{PD}_b \right)^2 . \tag{4}$$

However, if a bank's portfolio is strongly affected by overall business conditions the use of conditional *PD*s may be more appropriate:

$$min \sum_{b} \sum_{i \in b} \left(PD_{c,i} - \overline{PD}_{c,b} \right)^2 . \tag{5}$$

It may be supposed that banks grant higher loans to good borrowers than to borrowers with a relatively high *PD*. Thus, VaR that arises from a good borrower may be comparatively high, as well. Consequently, it might be more reasonable to use weighted versions of the objective functions (4) and (5) using the *EAD*s as weights, e.g., for the conditional *PD*s (5):

$$min \sum_{b} \sum_{i \in b} EAD_i \cdot \left(PD_{c,i} - \overline{PD}_{c,b} \right)^2 . \tag{6}$$

Moreover, banks may want to minimize the total absolute error between the regulatory capital computed with the true individual PD_is and the one computed in correspondence to the pooled *PD*s.

$$min \sum_{b} \sum_{i \in b} \left| RC\left(PD_i\right) - RC\left(\overline{PD}_b\right) \right| . \tag{7}$$

Apart from the selection of an appropriate objective function, several constraints imposed by the Basel II framework have to be taken into account when rating credit risk. First, according to paragraph 285 of the framework the pooled *PD* for corporate and bank exposures must be no smaller than

0.03%. Second, paragraphs 403 and 406 of the framework require banks to have a meaningful distribution of exposures without excessive concentrations. Thus, following Krink et al. (2007), we assume that no bucket may contain more than 35% of a bank's total exposure:

$$\frac{\sum_{i \in b} EAD_{b,i}}{\sum_b \sum_{i \in b} EAD_{b,i}} \leq 35\% \,. \tag{8}$$

Third, in order to avoid buckets that are too small, the number of borrowers in a bucket (N_b) should be larger than some percentage x of the entire number of borrowers N:

$$N_b \geq x \cdot N \,. \tag{9}$$

Again following Krink et al. (2007), we will assume $x = 1\%$ for our application in Section 5. However, we will define x based on statistical criteria when endogenizing the number of buckets in Section 6.

Fourth, the clustering algorithm must be set up such that buckets do not overlap and the union of buckets is the set of all borrowers. Furthermore, paragraph 404 of the framework requires banks to have at least seven borrower grades for non-defaulted borrowers.

4 An Optimization Heuristic for Credit Risk Bucketing

We tackle the *PD* bucketing problem as a clustering one, i.e., we want to determine the optimal partition of N bank clients in B buckets with respect to a given objective function and subject to some constraints (see Section 3). Since clustering problems are NP-hard when the number of clusters exceeds three (Brucker, 1978), stochastic search heuristics, such as Differential Evolution and Threshold Acceptance, can be a valid tool to tackle such problems. Furthermore, the presence of constraints narrows and segments further the search space. DE and TA allow to explore the whole search space, not focusing on the borders resulting from the constraints as conventional approaches often do. Following Krink et al. (2007), we build candidate solutions in TA or DE to encode the thresholds of buckets. Hence, when considering the problem in a continuous domain, the fitness landscape has large plateaus given that a change in the threshold of one bucket modifies the categorization only if there are some clients in the *PD* interval between the old and the

new thresholds, e.g, if a threshold varies from 0.2 to 0.21, the *PD*-bucketing partition would vary only if there are clients with *PD* in the interval [0.2, 0.21]. Then, the fitness value of each individual will vary across generation only when the new bucketing thresholds correspond to a new categorization. Given this inherent discrete nature of the problem, we expect TA to be a better alternative than DE. The reader is referred to Krink et al. (2007) for a description of Differential Evolution in general and for the credit risk bucketing problem in specific, while Threshold Acceptance is described in the following subsection.

4.1 Threshold Accepting

The idea of TA is to iteratively compare the objective function values of two candidate solutions that belong to the same neighborhood and to select one of them for further refinement. Thereby, the current candidate solution is replaced by a new one

- if this results in an improvement of the objective function value, or
- if a deterioration of the objective function value does not exceed a threshold as defined by a threshold sequence.

Due to the second feature, TA may overcome local optima.

TA requires to set an initial candidate solution and a criterion that terminates the search process. It turns out to be best to determine an initial candidate solution completely at random. Moreover, the search is stopped after a predetermined number of iterations. A nice feature of this stopping criterion is that the computation time can be controlled quite effectively.

In TA the current candidate solution is compared with a neighboring solution. Thus, the implementation requires to define a neighborhood structure. It is reasonable to define neighborhoods quite large at the beginning of the search but small towards its end. The idea underlying this procedure is to put more emphasis on exploring wide areas of the search space first but emphasizing a narrow search and refinement of a supposedly good candidate solution towards the end of the search.

Suppose the TA algorithm has generated for 7 buckets the starting solution $g_c = (3\%, 6\%, 10\%, 12\%, 17\%, 21\%)$ and *PD*s in our dataset are bound by the interval $[0.2\%; 24\%]$. Suppose further that the second bucket threshold is randomly selected for modification. The new candidate solution will be

a neighbor to the old one if the second bucket threshold is determined randomly from all *PD*s in the interval $[3\%; 10\%[$. The intervals for the remaining bucket thresholds can be found accordingly. The procedure is illustrated in Figure 1.

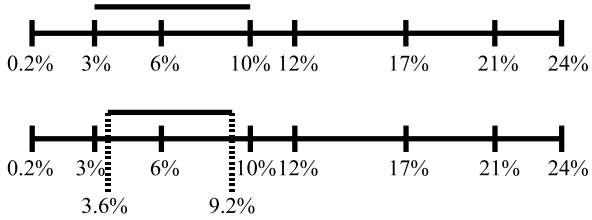

Figure 1: Bucket intervals.

As the search proceeds, these intervals shrink linearly in the current number of iterations relative to the total number of iterations. I.e., the contraction factor takes the form $[(I+1)-i]/I$. Consequently, after performing for example 20% of the iterations the second bucket threshold would be determined from the interval $[6\% - 0.8 \cdot (6\% - 3\%); 6\% + 0.8 \cdot (10\% - 6\%)[$.

New candidate solutions are generated from old ones by first determining randomly a bucket threshold of the current candidate solution and then replacing it with a random element from the above interval. This procedure is advantageous in at least two aspects. First, the objective function value of the new candidate solution g_n differs from the objective function value of the current candidate solution g_c only in the contribution of the two buckets that are affected by the alteration. Thus, fast updating of the objective function is feasible. Moreover, computation time becomes vastly independent of the number of buckets. This is due to the fact that for any number of buckets TA only has to compute the fitness of two buckets per iteration. On the contrary, in DE, as implemented in Krink et al. (2007), the fitness for all buckets is computed in every iteration. This results in a higher computation time. This disadvantage of DE becomes more pronounced for higher numbers of buckets, even if we are aware that different updating rules for DE similar to the ones employed for TA in the present application could speed up the runs. Second, since in TA new bucket thresholds are chosen from the *PD*s in the dataset, each new candidate solution constitutes a different partition and, consequently, a different value of the objective function which is not the case for our DE implementation on a continuous search space.

A final crucial element of any TA implementation is its threshold sequence since it determines TA's ability to overcome local optima. Basically, the idea is to accept g_n if its objective function value is better or if it is not much worse than that of g_c where *not much worse* means the deterioration may not exceed some threshold T defined by the threshold sequence.

We propose a threshold sequence that is based on the differences in the fitness of candidate solutions that are found in a certain area of the search space. Instead of using an ex ante simulation of local differences of the fitness function as proposed by Winker and Fang (1997), the local differences actually calculated during the optimization run are considered. By using a moving average, a smooth threshold sequence is obtained. Algorithm 1 provides the pseudocode for the TA implementation with the data driven generation of the threshold sequence.

Algorithm 1 Pseudocode for TA with data driven generation of threshold sequence.

Initialize I, $Ls = (0, \ldots, 0)$ of length 100
Generate at random an initial solution g_c, set $T = f(g_c)$
for $i = 1$ to I **do**
    Generate at random $g_n \in \mathcal{N}(g_c)$
    Delete first element of Ls
    if $f(g_n) - f(g_c) < 0$ **then**
        add $|f(g_n) - f(g_c)| \cdot (i/I)$ as last element to Ls
    else
        add $|f(g_n) - f(g_c)| \cdot (1 - i/I)$ as last element to Ls
    end if
    $T = \overline{Ls} \cdot (1 - i/I)$
    if $f(g_n) + T \leq f(g_c)$ **then**
        $g_c = g_n$
    end if
end for

The threshold sequence is calculated during the runtime of the algorithm and exhibits the following properties. First, it adapts to the region of the search space to which the current solution belongs. Second, it takes into account the current definition of the neighborhood. Third, and most importantly, it adapts to the objective function used. As a result, this data driven threshold sequence is readily available for use with any objective function, constraint handling technique or neighborhood structure and does not require any fine-tuning.

The current value of the threshold T is defined as the weighted mean $\overline{Ls}$ over the last 100 fitness differences (11:). A general requirement in TA is that thresholds should be larger at the beginning of the search in order to overcome local optima and decrease to zero at the end in order to reach at least a local, if not the global optimum. In order to satisfy this requirement, the weighted mean $\overline{Ls}$ is multiplied with a scaling factor decreasing linearly from one to zero with the number of iterations (11:).

Apart from this global weights, each fitness difference entering the vector Ls obtains a particular weight. At the beginning of the search process, one might expect many fitness improvements. For not being too generous in accepting deteriorations of the objective function, objective fitness differences corresponding to improvements are downweighted by the factor i/I, i.e., the share of iterations already done (7:). In contrast, towards the end of the search procedure, one has to expect that most trials result in a deterioration of the objective function. To avoid too generous thresholds, the corresponding elements of Ls are downweighted by the factor $(1-i/I)$ decreasing to zero with the number of iterations (9:). It is obvious that this threshold sequence adapts to the local structure of the search space. If the algorithm moves candidate solutions towards an optimum, fitness improvements are likely to become smaller the closer the algorithm approaches this optimum. Then, T declines and forces the algorithm not to deviate from its track towards the optimum. Once a (local) optimum is found only fitness deteriorations will be observed which increases T and eventually allows the algorithm to depart from that optimum and examine another part of the search space. By using a moving average, a smooth threshold sequence is obtained (11:).

4.2 Constraint Handling

When running the optimization heuristics TA and DE, the constraints described in Section 3 have to be taken into account. To this end, two alternative methods can be considered: rewriting the definition of domination, such that it includes the constraint handling (Deb et al., 2002) or imposing a penalty on infeasible solutions.

The first possibility has been described for the current application in Krink et al. (2007). The intuitive idea of this constraint handling technique is to leave the infeasible area of the search space as quickly as possible and never return. For minimization problems, the procedure can be described as follows within Algorithm 1:

1. If the new candidate solution g_n and the current candidate solution g_c satisfy the constraints, g_n replaces g_c if its fitness $f(g_n)$ satisfies the condition $f(g_n) + T \leq f(g_c)$. In TA T represents the threshold as defined by the threshold sequence. In DE, we set $T = 0$.

2. If only one candidate solution is feasible, select the feasible one.

3. If both solutions violate constraints, ...

   (a) ... select the one that violates fewer constraints.

   (b) ... if both solutions violate the same number of constraints, g_n replaces g_c if its fitness $f(g_n)$ satisfies the condition $f(g_n) + T \leq f(g_c)$. Again, T either takes a value as defined by the threshold sequence or, in DE, we set $T = 0$.

In contrast, the penalty technique allows infeasible candidate solutions while running the algorithm as a stepping stone to get closer to promising regions of the search space. In this case, the objective function is multiplied by a penalty term. Solutions should be penalized the stronger the more they violate the constraints. Moreover, in order to guarantee a feasible solution at the end, the penalty should increase over the runtime of the algorithm. Equation (10) states that the objective function value f_u of a candidate solution is increased by some penalty factor $A \in [1; 2]$ that puts more weight on penalties the more the current iteration i approaches the overall number of iterations I. The exponent a may take values in the interval $[0; 1]$. No penalty is placed on f_u if no constraint is violated so that $a = 0$. However, if the constraints are violated most strongly, i.e., all borrowers are concentrated in one bucket leaving the remaining buckets empty, the exponent takes the value $a = 1$. A more formal description of this penalty technique is given in the appendix.

$$f_c(g) = f_u(g) \cdot A = f_u(g) \cdot \left(1 + \sqrt{\frac{i}{I}}\right)^a . \tag{10}$$

For the current application, DE is only implemented with the constraint-dominated handling technique, while for TA both methods are implemented. Generally, the constraint-dominated handling technique performs well while taking comparatively little computation time. However, depending on the kind of objective function used the penalty technique may improve the reliability of TA, i.e., reduce the variance of the results obtained.

5 Results and Relative Performance

For our empirical application we consider the dataset comprising 11995 defaulted and non-defaulted borrowers of a major Italian bank already analyzed by Krink et al. (2007). The *PD*s are point in time and range between 0.21% and 23.94%. Moreover, the conditional probability of default (PD_c) was computed using equation (2). The PD_cs range between 4.52% and 64.88%.

All algorithms are implemented from scratch in Matlab 7.6 and run on a PC with Intel Duo Core processor operating at 2.40 GHz and running Windows XP. Please notice that the reported empirical results are slightly different from the ones reported by Krink et al. (2007) due to the fact that we use our own implementation of Differential Evolution.

5.1 Results for Fixed Number of Buckets

Tables 1 to 3 report the empirical results of the two heuristic algorithms for 7, 10, and 15 buckets, the two different constraint handling techniques and the three objective functions described above. Both algorithms were restarted 30 times on each problem instance to control for the stochasticity of heuristic optimization techniques. For the comparison of the two methods, we report the best value, the median, the worst value, the variance, the 80% percentile, the 90% percentile, and the frequency the best value occurs in all 30 repetitions.

The tuning parameters of the DE implementation are the scaling factor F and the crossover probability CR. These settings might affect the quality of results depending on the properties of the problem. To determine the parameter values that result in the best objective function values, we run the algorithm 30 times for different combinations of F and CR, both ranging between 0.5 and 0.9. Thereby, the objective function (6) was used. The distribution of the results indicates that, for the specific problem instance, tuning the technical parameters does not affect the solution quality for values of CR larger than 0.6, while the choice of F within the given interval appears to be irrelevant. Results are available upon request.

Then, since extensive parameter tuning on DE suggests that DE is rather insensitive to the choice of F and CR, we fix the initial parameter settings such that the population size is $n_p = 100$ and the number of generations is $n_G = 1\,000$, while the scaling factor F and the crossover rate CR were kept constant at 0.5 and 0.8, respectively. In the case of TA, the algorithm

Table 1: Objective function (5)

	Best	**Mean**	**Worst**	**s.d.**	**q80%**	**q90%**	**Freq**
				$B = 7$			
TA[a]	5.9184	5.9184	5.9184	0.0000	5.9184	5.9184	20/30
TA[b]	5.9184	5.9184	5.9184	0.0000	5.9184	5.9184	30/30
DE	5.9184	5.9211	5.9223	0.0018	5.9223	5.9223	9/30
				$B = 10$			
TA[a]	3.9155	3.9226	3.9369	0.0101	3.9366	3.9366	18/30
TA[b]	3.9155	3.9190	3.9366	0.0080	3.9155	3.9366	19/30
DE	3.9155	9.9319	4.1663	0.0496	3.9195	3.9527	2/30
				$B = 15$			
TA[a]	2.8842	2.8848	2.8929	0.0016	2.8855	2.8855	6/30
TA[b]	2.8842	2.8874	2.9064	0.0053	2.8929	2.8933	7/30
DE	2.8964	2.9761	3.0199	0.0428	3.0083	3.0140	1/30

[a]Rejection based constraint handling technique
[b]Penalty technique

Table 2: Objective function (6) in EUR

	Best	**Mean**	**Worst**	**s.d.**	**q80%**	**q90%**	**Freq**
				$B = 7$			
TA[a]	4,582.86	4,582.86	4,582.86	0.0000	4,582.86	4,582.86	30/30
TA[b]	4,582.86	4,582.86	4,582.86	0.0000	4,582.86	4,582.86	30/30
DE	4,582.86	4,587.35	4.671.38	16.5569	4,583.52	4,585.09	2/30
				$B = 10$			
TA[a]	3,471.68	3,479.97	3,483.92	1.8592	3,480.01	3,480.21	1/30
TA[b]	3,471.68	3,480.33	3,483.92	2.8705	3,483.66	3,483.92	2/30
DE	3,471.51	3,475.47	3,498.96	5.4891	3,479.96	3,480.18	4/30
				$B = 15$			
TA[a]	2,821.00	2,833.55	2,865.98	14.1177	2,844.24	2,856.92	8/30
TA[b]	2,821.00	2,830.55	2,860.02	11.5971	2,840.99	2,844.24	3/30
DE	2,866.23	2,943.09	3,122.39	57.9240	2,958.48	3,005.19	1/30

[a]Rejection based constraint handling technique
[b]Penalty technique

was run for $I = 100\,000$ iterations, in order to attain analogy with DE's population size and number of generations. It should be noted that due to the local updating method in TA, the 100 000 iterations of TA require less computing time than the corresponding run of DE. The relative merits of both methods in terms of computational load and quality of results are reported in Section 5.2.

Table 1 presents a statistical summary of the results using objective function (5). The TA algorithm was run using both the rejection based constraint handling technique and the penalty technique. The results are affected by the choice of the constraint handling technique, as the best value is obtained with an equal or higher frequency when using the penalty technique. However, these results cannot be generalized for the alternative objective functions (6) and (7) (see Tables 2 and 3), especially for a larger number of buckets, i.e., $B = 10, 15$. In general, the performance of the TA implementation is excellent for the case of seven buckets and still gives good results with low variance for the larger problem instances.

Table 3: Objective function (7) in EUR

	Best	**Mean**	**Worst**	**s.d.**	**q80%**	**q90%**	**Freq**
				B = 7			
TA[a]	45,791.49	45,793.18	45,825.62	6.4870	45,791.49	45,791.72	26/30
TA[b]	45,791.49	45,794.11	45,826.57	7.7787	45,791.49	45,801.65	25/30
DE	45,791.49	45,810.09	46,004.19	39.4151	45,828.39	45,828.39	4/30
				B = 10			
TA[a]	31,942.19	31,951.30	31,996.89	19.9770	31,946.42	31,994.72	21/30
TA[b]	31,942.19	31,951.25	31,994.73	18.8050	31,946.42	31,992.74	18/30
DE	31,995.28	32.166.86	32.299.00	119.9837	32.299.00	32,299.00	1/30
				B = 15			
TA[a]	20,711.93	20,729.26	20,973.53	62.6797	20,713.36	20,714.88	10/30
TA[b]	20,711.93	20,725.99	20,951.01	49.5889	20,714.88	20,715.61	1/30
DE	20,970.37	24,916.30	35,003.82	4875.7503	31,215.84	33,676.88	1/30

[a] rejection based constraint handling technique
[b] penalty technique

Considering the performance of the DE implementation, we observe that the best value is obtained for $B = 7$ at a frequency of 9 out of 30 restarts.

While, for a higher number of buckets, the best value does not deviate much from the optimum, the efficiency worsens. The same pattern is observed for all three objective functions.

We conclude that the TA implementation is superior for most problem instances in terms of mean solution quality and variance for all objective functions considered. The clustering of credit risk is a problem on a discrete search space. In contrast to the DE algorithm, the TA implementation takes this discrete feature of the search space into account. This might explain its superior performance.

5.2 Relative Performance of DE and TA

Section 5.1 provides evidence of the good performance of both algorithms for the credit risk bucketing problem. Given that TA exploits the discrete structure of the search space and uses a local updating procedure, it is significantly faster than DE for a given number of function evaluations.

Therefore, in order to obtain a fair comparison of both algorithms, we consider two settings. First, we analyze the distribution of results obtained from both algorithms when running them for the same time. Second, we fix a quality goal, e.g., not to deviate by more than 1% from the best solution documented above. Then, both algorithms are run using increasing computational time until at least 50% of the restarts meet the quality goal.

For the first approach, the following setup is used. We run the DE algorithm with the same parameters as above, i.e., population size $n_p = 100$ and number of generations $n_G = 1000$, and – to have a comparison for a small amount of computational resources – with $n_p = 40$ and $n_G = 50$. Then, we estimate the number of iterations I which can be performed by our TA algorithm using the same computational time. In fact, this number of iterations will depend on the objective function used and on the number of buckets B, as the advantage of updating becomes more pronounced for larger B.

Table 4 summarizes the findings. The first four columns report respectively the objective function, the number of buckets B, the population size n_p and the number of generations n_G of DE. Column (5) displays the computing time for a single restart of our implementations. Column (6) reports the number of iterations I in TA that equalizes computation time for DE and TA. Columns (7) and (8) display the difference in the mean and standard deviation between TA and DE. Thereby, negative values indicate an advantage of the TA implementation. Finally, column (9) reports the number of

times, TA outperforms DE.

Table 4: Relative performance of DE and TA for given computing time

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
				CPU	I	Δ	Δ	**better result**	
Obj.	B	n_P	n_G	**time**	**for TA**	**mean**	**Std.**	**for TA**	**for DE**
(7)	7	100	1000	32.4m	786 600	-18.4	-38.3	25/30	0/30
	10	100	1000	36.0m	1 050 000	-220.7	-108.0	21/30	0/30
	15	100	1000	192.7m	6 760 511	-4202.8	-4873.9	5/30	0/30
(7)	7	40	50	42.3s	17 000	-13 145	-10 210	6/30	0/30
	10	40	50	51.7s	25 000	-16 355	-8 630	6/30	0/30
	15	40	50	66.7s	38 000	-23 441	-64 880	1/30	0/30

Table 4 reports results for objective function (7) and $B = 7, 10, 15$ buckets. When considering the original setting for DE with $n_P = 100$ and $n_G = 1000$, the number of iterations for TA can be increased above the value of 100 000 given the same computation time. This further increase in the number of iterations does not affect the quality of results. However, when considering a smaller amount of available computational time, e.g., $n_P = 40$ and $n_G = 50$, it becomes obvious that TA still outperforms DE when using the same computational time. Furthermore, the standard deviation is drastically reduced for TA.

For the second approach mentioned above, we consider three quality levels, i.e., 10%, 5% and 1% departure from the optimum values reported. Taking into account computation time, we only report findings for objective function (6) and $B = 7$ and $B = 15$ buckets, respectively. For DE we fix $n_p = 100$ and increase the number of generations n_G, while for TA the number of iterations I varies. For both algorithms the parameter (n_G or I) is increased stepwise until the algorithm finds a solution meeting the quality level in at least 50% out of 30 replications. The results are summarized in Table 5 providing both the parameters actually used for the two algorithms and the corresponding CPU times for a single restart.

It is evident that a given quality of the solutions can be obtained much faster with TA. The relative advantage becomes even more pronounced for the larger problem instance ($B = 15$). This effect is due to the local updating used with the TA algorithm. In fact, for $B = 15$, the quality goal of 1% from

Table 5: Timing of DE and TA for given solution quality

			DE		TA	
Precision	B	n_P	n_G	**time**	I	**time**
10%	7	100	30	44.44s	200	0.62s
5%			40	58.45s	300	0.72s
1%			70	104.24s	800	1.27s
10%	15	100	100	5.19m	2 000	0.03m
5%			1000	45.75m	3 000	0.05m
1%			$*$	$*$	15 000	0.21m

$*$: No solution obtained for $n_G \leq 5000$ generations.

the best value could not be satisfied in at least 50% of the cases by DE even when using $n_G = 5000$ generations. For this parameter setting, a single run of the DE algorithm takes more than 4 hours of CPU time. By contrast, the same quality goal can be obtained by the TA algorithm in less than a minute.

6 Endogenous Determination of Number of Buckets

The Basel II framework requires banks to have a *meaningful* credit risk rating system. This does not only refer to the clustering of clients into a given number of *PD*-buckets, but also to the choice of the number of buckets. Thereby, a trade-off has to be faced. On the one hand, the clusters of borrowers should be rather homogenous. Increasing the number of buckets will reduce the loss in precision that comes from replacing individual *PD*s with pooled *PD*s. This effect causes objective function values to decline as the number of buckets is raised, resulting in a larger optimum number of *PD*-buckets.

On the other hand, both banks and regulators will be interested in an ex post validation of the classification system. For example, one may want to evaluate ex post if the observed number of defaults matches the ones predicted by the credit risk rating system. In this context, looking at the number of defaults may be seen as a proxy for evaluating whether the credit risk rating system will predict unexpected losses correctly, which in turn,

results in a statement about the adequacy of banks' regulatory capital. Alternatively, one might consider directly the precision of the estimates of unexpected losses. A crucial factor driving the precision of any ex post evaluation is the number of borrowers per bucket. Thus, imposing a requirement on the minimum number of borrowers in a bucket based on ideas of ex post validation will result in an optimum (maximum) number of buckets still satisfying this constraint.

6.1 Validating Unexpected Losses

Ex post validation may be based on the correct statement of unexpected losses (UL), respectively regulatory capital since $RC = 1.06 \cdot UL$. Supervisory authorities' objective is to motivate banks to set aside equity capital equaling at least 8% of their risk-weighted assets in order to ensure the stability of the banking system. On the contrary, the objective of profit maximization requires banks' to back up their risk-weighted assets with no more than the supervisory authority's minimum requirements. These objectives can be operationalized by stating that in no bucket b actual unexpected losses in a stress-situation ($UL_{b,a}$) shall be smaller or larger than predicted unexpected losses (UL_b) plus or minus some fraction ε of bucket b's stake in total unexpected losses as measured by the percentage of its borrowers (N_b) in the number of all borrowers (N) (see equation (11)). Total unexpected losses, unexpected losses of bucket b and pooled conditional probabilities of default are given by equations (12), (13), and (14), respectively.

$$UL_b - \varepsilon \cdot \left(UL \cdot \frac{N_b}{N} \right) \leq UL_{b,a} \leq UL_b + \varepsilon \cdot \left(UL \cdot \frac{N_b}{N} \right) \tag{11}$$

$$UL = 0.45 \cdot \sum \left(EAD_i \cdot (PD_{c,i} - PD_i) \right) \tag{12}$$

$$UL_b = 0.45 \cdot \sum_{i \in b} \left(EAD_i \cdot (PD_{c,b} - PD_b) \right) \tag{13}$$

$$PD_{c,b} = \frac{\sum_{i \in b} PD_{c,i}}{N_b} \,. \tag{14}$$

Equation (11) is not operational since we do not know the distribution of unexpected losses. Given that we know the distribution of defaults, we can approximate $UL_{b,a}$ by $N_b \cdot \overline{UL}_b$. Then, dividing equation (11) by $\overline{UL}_b$ and multiplying it with $\overline{PD}_b$, we obtain equation (15). This equation can be

interpreted meaningfully as well since it says that for condition (11) to hold the actual number of defaults in bucket b ($\tilde{D}_b$) must lie within an interval $[D_{b,min}; D_{b,max}]$. The size of this interval is determined by several parameters. Obviously, it increases with the expected probability of default and the number of borrowers in bucket b. Moreover, it rises with ε. Finally, the interval becomes larger, and thus easier to satisfy, if the mean unexpected loss in bucket b ($\overline{UL}_b$) is smaller than the mean of total unexpected loss ($\overline{UL}$), i.e., the default of a borrower in this bucket is less likely to endanger the bank's stability than an average borrower's default. On the other hand, the interval shrinks and thus becomes harder to satisfy if borrowers are likely to cause an above average unexpected loss.

$$D_{b,min} \leq \tilde{D}_b \leq D_{b,max}$$

$$N_b \cdot PD_b \cdot \left[1 - \varepsilon \cdot \frac{\overline{UL}}{\overline{UL}_b}\right] \leq \tilde{D}_b \leq N_b \cdot PD_b \cdot \left[1 + \varepsilon \cdot \frac{\overline{UL}}{\overline{UL}_b}\right] . \qquad (15)$$

The central idea of this approach is to have a sufficient number of borrowers in each bucket so that we can ex ante state with a certain confidence $1 - \alpha$ that the actual number of defaults should lie within the interval $[D_{b,min}; D_{b,max}]$. Since the actual default for a loan is a binary variable, the number of actual defaults within a bucket can be modeled by the binomial distribution (see Hunt et al. (2009) for using the beta-binomial distribution). Consequently, a $1 - \alpha$ confidence interval for $\tilde{D}_b$ is defined by:

$$\begin{aligned} P_{int} &= P_b\left(D_{b,min} \leq \tilde{D}_b \leq D_{b,max}\right) \\ &= \sum_{k=D_{b,min}}^{D_{b,max}} \binom{N_b}{k} \cdot \overline{PD}_b^k \cdot \left(1 - \overline{PD}_b\right)^{N_b - k} \geq 1 - \alpha \,. \end{aligned} \qquad (16)$$

For equation (16) to hold we assume that the default risks are independent, which is unrealistic. Further research will be devoted to relax this assumption and deepen our analysis.

Given a bucket b of size N_b, we just have to check whether the constraint $P_{int} \geq 1 - \alpha$ is satisfied. Thus, our requirement on the precision of ex post validation imposes an additional constraint to the optimization problem. For the consideration of this additional constraint in the penalty term, the reader is referred to the details provided in the appendix.

Using this concept, we define a credit classification system as *meaningful* if it allows for an ex post validation at a given level of precision as described by the two parameters α and ε. The sample composition, in particular the total sample size, and bank objectives will affect the choice of ε.

Constructing buckets as previously described allows to easily validate the accuracy (as determined by the choice of ε) of a bank's credit risk rating system. If we find the actual number of defaults in any bucket b to lie outside the interval $[D_{b,min}; D_{b,max}]$ we can state with confidence $1-\alpha$ that the credit risk rating system is not suitable for predicting defaults in that bucket. This may have at least two reasons. First, the objective function that is used for partitioning the dataset may not be appropriate. One may easily check for this problem by using different objective functions and then assessing which ones yield results that do not cause actual defaults to lie outside the above bounds. Second, the bank employs a statistical default prediction model that does not forecast defaults correctly and thus needs to be improved.

6.2 Results for Endogenous Number of Buckets

In this section we evaluate the quality of the *UL*-constraint proposed in Section 6.1. The results are obtained from running TA 30 times with 200 000 iterations. We evaluate objective functions (5) based on squared differences of PD_cs, (6) based on weighted squared differences of PD_cs, and (7) based on differences in RC in absolute terms. We fix LGD_i equal to 0.45 for all bank clients.

One should note that not all combinations of α and ε are feasible for a given total number of loans and taking into account the other constraints imposed by the Basel II framework. Searching over a grid of different values for α and ε (see Appendix) we find $\alpha = 10\%$ and $\varepsilon = 30\%$ to be a good choice of parameters. This combination gives a sufficient level of confidence, a reasonable maximum number of buckets and allows for a sensible interpretation of the interval $[D_{b,min}; D_{b,max}]$. Please note that the higher the value we choose for α the larger will be the risk of a β-error, i.e., accepting $\overline{PD_b}$ as an unbiased estimator of bucket b's true default rate while it is not.

Thus, if we find ex post the actual number of defaults in all buckets to lie in the interval defined by equation (15) we can state with 90% confidence that actual unexpected losses do not deviate from unexpected losses predicted by the credit risk rating system by more than $\pm 30\%$ of the buckets' fraction (as measured by the number of borrowers) in total unexpected loss. Taking into

account the small size of our sample (11 995 borrowers) we are confident that these values can be improved drastically for larger samples.

Using the *UL*-constraint gives similar results for objective functions (5) and (6) such that stylized facts on these functions can be presented together. Please note that without using the *UL*-constraint objective functions (5) and (6) produce quite dissimilar results, i.e., objective function (5) places a sizeable amount of borrowers in the first bucket while objective function (6) produces a more evenly distribution of borrowers across buckets.

1. The first results indicate that the best number of buckets is between 10 (for objective function (6) and 12 (for objective function (5)).

2. When increasing the number of buckets, the algorithm does not always find a feasible solution. In fact, the *UL*-constraint makes the optimization problem more complex by narrowing the search space even more.

3. For a seven bucket setting an idealized solution-vector of buckets' mean *PD*s looks like $g_s = (0.25\%; 0.55\%; 1.5\%; 4\%; 8\%; 14\%; 21\%)$. The *UL*-constraint shapes the solution in a way that we must not reject the validity of the credit risk rating system if we find ex post actual $\overline{PD_b}s$ that deviate from predicted $\overline{PD_b}s$ by less than $\pm$ the allowed deviations (in percentage points) given by $d = (0.2\%; 0.25\%; 0.4\%; 1\%; 1.8\%; 3.5\%; 6.5\%)$.

   (a) We find that the *UL*-constraint imposes constraints on mean *PD*s that are of a reasonable size.

   (b) The constraint on the first bucket is quite generous since it contains good borrowers that are unlikely to default.

   (c) It is restrictive for mid-range borrowers allowing actual mean *PD*s to only deviate from predicted mean *PD*s by roughly 1/4. This is reasonable since it is highly uncertain whether these borrowers will default and cause a high unexpected risk for the bank.

   (d) The *UL*-constraint becomes more generous for the last bucket again, allowing actual mean *PD*s to deviate from predicted mean *PD*s by roughly 1/3. This is reasonable since these borrowers' default is quite likely such that high provisions have already been recognized. Hence, a smaller portion of their default risk must be backed up with capital requirements.

4. The rejection based constraint handling technique gives us better results (i.e., better objective function values and fewer runs converging to an infeasible solution) than the penalty technique.

Objective function (7) gives slightly different results since it allocates borrowers more evenly and especially puts less borrowers in the first bucket. We find the idealized vectors $g_s = (0.23\%; 0.3\%; 0.6\%; 0.9\%; 3\%; 7\%; 18\%)$ and $d = (0.2\%; 0.2\%; 0.25\%; 0.25\%; 1\%; 2\%; 5\%)$ using the terminology introduced before.

Summarizing, we find that the structure of results when using the *UL*-constraint is quite reasonable. It puts more emphasis on critical, i.e., mid-range borrowers and yields intervals around mean *PD*s that reflect the structure of borrowers. Moreover, imposing the *UL*-constraint somewhat increases the computational burden by narrowing down the search space. As a consequence, for 200 000 iterations the TA optimization heuristic converges towards different solutions in repeated runs. A nice feature of the *UL*-constraint, even for our small dataset, is to give us feasible solutions for reasonable values of α and ε. This enables us to test our validation-hypothesis. Thus, for a larger number of borrowers results may be expected to improve massively.

7 Conclusion

The Basel II capital accord requires banks to group loans according to their creditworthiness and set aside equity in order to self-insure against unexpected losses from borrowers' defaults that occur under sufficiently negative economic conditions. Previous work has shown that this task can be tackled as a clustering problem, where the objective is to minimize the loss in precision, which inevitably occurs when borrowers in the same bucket are assigned the same probability of default. Furthermore, real-world constraints can increase the complexity of the optimization problem. Optimization heuristics can then be a reliable and viable tool to use. In this work, we extend previous research in two directions.

First, we suggest to use the Threshold Accepting algorithm and show that this approach allows to minimize the loss in precision more effectively, more reliably, and more efficiently than Differential Evolution. TA finds partitions that have a smaller loss in precision than those found by DE. TA converges to better grouping solutions in less computational time.

Second, we propose an approach for determining the optimal number of buckets. To our knowledge, this topic has not been addressed in the literature before, although it is of great importance for practitioners. We aim to tackle the problem by designing a bank's credit rating system such that its quality may be validated ex-post. The loss in precision by grouping borrowers together rather than treating them as individuals decreases as the number of buckets increases. Moreover, banks and regulatory authorities are concerned with stating regulatory capital (respectively unexpected losses) correctly. Thus, we propose to cluster borrowers such that we may evaluate ex post with a given confidence level whether actual unexpected losses fall within a sufficiently narrow interval around predicted unexpected losses. Then, the optimal number of buckets is the maximum number of buckets that allows us to support our statement with a given confidence level. Our evaluations of this constraint suggest that it influences the structure of clusters in a reasonable way. Moreover, we find that even for small sample sizes it allows us to use up to eleven buckets for reasonable confidence- and precision-levels.

We show that our approach can provide meaningful insights into the problem of determining the optimal structure of *PD* buckets. However, we are aware that further research and empirical investigation on different loss functions and larger real-world datasets is required. Moreover, it is of special interest which confidence- and precision-levels may be used for different sample sizes. In this context, also more realistic assumptions about the dependency structure of unexpected losses in a credit portfolio might be considered. Finally, although the constraint imposed on unexpected losses has a strong theoretical support, one might also consider alternative formulations or approximations resulting in a lower computational complexity for calculating the constraints. Thereby, the efficiency of the algorithm could be improved even further.

8 Acknowledgements

We are thankful to Andrea Resti, two anonymous referees and the Editor for their helpful advices; we also thank Giovanni Butera (Moodys KMV) for kindly providing the data. Sandra Paterlini conducted part of this research while visiting the School of Mathematics, University of Minnesota. Financial support from the EU Commission through MRTN-CT-2006-034270 COMISEF, from MIUR PRIN 20077P5AWA005 and from Fondazione Cassa

di Risparmio di Modena for ASBE Project is gratefully acknowledged.

Appendix

Penalty Term

The exponent a used in the penalty term (10) is defined as follows:

$$a = \left(0.5 \cdot \sum_b D_{EAD,b} \cdot \frac{\sum_{i \in b} EAD_{b,i} - 35\% \cdot \sum_b \sum_{i \in b} EAD_{b,i}}{65\% \cdot \sum_b \sum_{i \in b} EAD_{b,i}} \right) + \left(0.5 \cdot \frac{\sum_b D_{N,b} \cdot \frac{x \cdot N - N_b}{x \cdot N}}{\sum_b D_{N,b}} \right) . \quad (17)$$

The idea of the penalty technique is to allow infeasible candidate solutions while running the algorithm as a stepping stone to get closer to promising regions of the search space. In this case, a penalty is multiplied on the objective function value that depends on the extent of constraint violations. In order to guarantee a feasible solution at the end, this penalty should increase over the runtime of the algorithm. The problem-specific penalty weights used in our application are defined by equations (10) and (17). They state that the objective function value f_u of a candidate solution is increased by some penalty factor $A \in [1; 2]$ that puts more weight on penalties the more the current iteration i approaches the overall number of iterations I. No penalty is placed on f_u if no constraint is violated so that $a = 0$. However, the variable a may take values up to 1 if the violation of the constraints reaches its maximum value. If the sum of EAD in some bucket b exceeds 35% of total EAD $D_{EAD,D}$ takes value 1 and 0 otherwise. $D_{N,b}$ takes value 1 if bucket b contains less than 1% of all borrowers. Both binding constraints are equally weighted.

Confidence Interval

Let us define the dummy variable $D_{N,b}$, which takes the value 1 if the constraint is violated and 0 otherwise. When constraints are considered based on rejection of infeasible candidate solutions the algorithms described above will not change. However, if the penalty technique is used it is necessary to

alter equation (17) by removing the second summand in (17) and adding a term for the degree of violation of the additional constraint as exhibited by the second term in (18):

$$a = \left(0.5 \cdot \sum_b \cdots\right) + \left(0.5 \cdot \frac{\sum_b D_{N.b} \cdot \frac{1-\alpha-P_{int}}{1-\alpha}}{\sum_b D_{N,b}}\right). \tag{18}$$

Results with Ex Post Validation

In the following, the numerical results shall be presented that are discussed and interpreted in Section 6. In this section we evaluate the quality of the *UL*-constraint. The results were obtained from running TA 30 times with 200 000 iterations. We evaluate objective functions (5) based on squared differences of PD_cs, (6) based on weighted squared differences of PD_cs, and (7) based on differences in RC in absolute terms. We choose $\alpha = 10\%$ and $\varepsilon = 30\%$. The last column gives the number of runs that converge towards the best solution relative to all runs that produce a solution meeting all constraints. For the problem instances, for which no feasible solution could be found in 30 runs, we report "n.a." in the corresponding cells of the tables.

Grid Search

In order to find meaningful values for α and ε we run the TA algorithm 15 times with 200 000 iterations for $\alpha = (5\%, 10\%, 15\%)$, ε=(20%, 25%, 30%, 35%, 40%) and 7 to 15 buckets. Evaluations are done using the rejection based constraint handling technique which we find to give better results than the penalty technique. Evaluations are done for objective function (5)). The results are shown in table 9, which reports the maximum number of buckets b where feasible solutions are found. Moreover, it gives the number of runs (out of 15 runs) that converged to a feasible solution when using this maximum number of buckets.

The results indicate that $\alpha = 5\%$ is quite restrictive and requires relatively large values of ε while $\alpha = 15\%$ is larger than standard confidence levels (i.e., 5% and 10%). Moreover, $\alpha = 15\%$ appears to be overly generous so that $\alpha = 10\%$ is a good choice. Since for $\alpha = 10\%$, $\varepsilon = 30\%$ is the only value that gives a maximum number of buckets within the reasonable bucket range (i.e., 7 to 15 buckets) it is selected as parameter value for further evaluations.

Table 6: Objective function (5) in EUR

	Best	**Mean**	**Worst**	**s.d.**	**q80%**	**q90%**	**Freq**
				$B = 7$			
TA[a]	5.9757	6.0164	6.0946	0.0273	6.0324	6.0606	1/30
TA[b]	6.0028	6.1336	6.2236	0.0568	6.1781	6.1985	1/30
				$B = 8$			
TA[a]	5.2325	5.5768	6.8232	0.2999	5.6743	5.7408	1/30
TA[b]	5.4304	5.7657	6.0490	0.1405	5.8737	5.9186	1/30
				$B = 9$			
TA[a]	5.0450	5.8765	7.1784	0.5355	6.4458	6.6844	1/30
TA[b]	5.3768	5.8476	6.2477	0.2487	6.0762	6.1363	1/30
				$B = 10$			
TA[a]	4.9733	5.7567	8.2735	0.708	6.0658	6.5098	1/25
TA[b]	5.4881	6.1514	7.0235	0.41826	6.4237	6.6459	1/25
				$B = 11$			
TA[a]	5.0589	5.6022	6.6355	0.5143	5.8587	6.3196	1/13
TA[b]	5.9513	8.7813	19.51	5.9981	6.1685	19.51	1/5
				$B = 12$			
TA[a]	4.6789	4.6789	4.6789	0.0000	4.6789	4.6789	1/1
TA[b]	7.7075	7.7075	7.7075	0.0000	7.7075	7.7075	1/1
				$B = 13$			
TA[a]	5.722	6.0172	6.3125	0.4176	6.3125	6.3125	1/2
TA[b]	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	0/0

[a]Rejection based constraint handling technique
[b]Penalty technique

Table 7: Objective function (6) in EUR

	Best	**Mean**	**Worst**	**s.d.**	**q80%**	**q90%**	**Freq**
				$B = 7$			
TA[a]	4,722.39	4,880.31	6,175.71	264.77	4,893.71	5,027.66	1/30
TA[b]	4,905.15	5,169.61	5,325.29	107.53	5,255.33	5,272.40	1/30
				$B = 8$			
TA[a]	4,582.44	4,957.22	6,299.73	390.81	5,036.84	5,431.29	1/30
TA[b]	4,859.07	5,112.88	5,477.91	161.00	5,237.02	5,327.58	1/30
				$B = 9$			
TA[a]	4,573.68	5,203.98	6,693.43	569.72	5,431.46	6,312.35	1/30
TA[b]	4,749.35	5,388.93	6,187.62	301.73	5,640.82	5,735.11	1/29
				$B = 10$			
TA[a]	4,426.02	4,770.40	5,116.55	190.36	4,904.38	5,025.04	1/11
TA[b]	5,330.69	6,211.82	8,367.79	1,022.38	7,015.54	7,431.06	1/11
				$B = 11$			
TA[a]	5,068.02	5,068.02	5,068.02	0.00	5,068.02	5,068.02	1/1
TA[b]	5,081.86	5,376.79	5,671.71	417.09	5,671.71	5,671.71	1/2
				$B = 12$			
TA[a]	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	0/0
TA[b]	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	0/0
				$B = 13$			
TA[a]	6,473.01	6,473.01	6,473.01	0.00	6,473.01	6,473.01	1/1
TA[b]	6,573.46	6,573.46	6,573.46	0.00	6,573.46	6,573.46	1/1

[a]Rejection based constraint handling technique
[b]Penalty technique

Table 8: Objective function (7) in EUR

	Best	**Mean**	**Worst**	**s.d.**	**q80%**	**q90%**	**Freq**
				$B = 7$			
TA[a]	52,188.31	55,366.40	57,103.79	809.19	55,779.96	55,897.19	1/30
TA[b]	55,334.48	59,123.42	63,163.35	2,144.92	60,799.91	61,600.04	1/30
				$B = 8$			
TA[a]	49,608.70	53,181.84	57,558.35	2,326.82	55,048.37	55,073.45	1/27
TA[b]	55,769.95	62,561.90	68,083.51	2965.40	64,448.67	65,081.21	1/18
				$B = 9$			
TA[a]	47,053.88	51,589.92	54,441.70	3,025.26	54,078.58	54,078.58	1/6
TA[b]	55,140.37	58,516.37	61,387.41	2,592.12	60,652.09	61,387.41	1/5
				$B = 10$			
TA[a]	48,106.88	50,195.77	52,284.67	2,954.15	52,284.67	52,284.67	1/2
TA[b]	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	0/0
				$B = 11$			
TA[a]	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	0/0
TA[b]	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	0/0
				$B = 12$			
TA[a]	n.a.	n.a.	n.a.	n.a.	n.a.	n.a.	0/0
TA[b]	94,083.11	94,083.11	94,083.11	0.00	94,083.11	94,083.11	1/1

[a]Rejection based constraint handling technique
[b]Penalty technique

Table 9: Evaluation of objective (5) for combinations of α and ε

α \ ε	20%		25%		30%		35%		40%	
	max b	feas. runs	max b	feas. runs	max b	feas. runs	max b	feas. runs	max b	feas. runs
5%	<7	0/15	<7	0/15	<7	0/15	10	2/15	14	1/14
10%	<7	0/15	7	4/15	14	1/14	15	1/15	15	15/15
15%	<7	0/15	12	2/15	15	6/15	15	15/15	15	15/15

References

Basel Committee on Banking Supervision, 2005. An explanatory note on the Basel II IRB Risk Weight Functions. Tech. rep., Bank for International Settlements.

Brucker, P., 1978. On the complexity of clustering problems. In: Beckmenn, M., Kunzi, H. (Eds.), Optimisation and Operations Research. Vol. 157 of Lecture Notes in Economics and Mathematical Systems. Springer, Berlin, pp. 45–54.

Deb, K., Pratap, A., Agarwal, S., Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm. IEEE Transactions on Evolutionary Computation 6 (2), 182–197.

Dueck, G., Scheuer, T., 1990. Threshold Accepting: A general purpose algorithm appearing superior to Simulated Annealing. Journal of Computational Physics 90, 161–175.

Foglia, S., Iannotti, P., Reedtz, P. M., 2001. The definition of the grading scales in banks' internal rating systems. Economic Notes 30 (3), 421–456.

Gilli, M., Winker, P., 2004. Applications of optimization heuristics to estimation and modelling problems. Computational Statistics & Data Analysis 47, 211–223.

Hunt, D. L., Chenga, C., Poundsr, S., 2009. The beta-binomial distribution for estimating the number of false rejections in microarray gene expression studies. Computational Statistics & Data Analysis 53, 1688–1700.

Kiefer, N., Larson, C., 2004. Evaluating design choices in economic capital modeling: A loss function approach. Cornell University; Office of the Controller of the Currency.

Krink, T., Paterlini, S., Resti, A., 2007. Using differential evolution to improve the accuracy of bank rating systems. Computational Statistics & Data Analysis 52 (1), 68–87.

Krink, T., Paterlini, S., Resti, A., 2008. The optimal structure of PD buckets. Journal of Banking and Finance 32 (10), 421–456.

Paterlini, S., Krink, T., 2006. Differential evolution and particle swarm optimization in partitional clustering. Computational Statistics & Data Analysis 50 (5), 1220–1247.

Winker, P., 2001. Optimization Heuristics in Econometrics: Applications of Threshold Accepting. Wiley, Chichester.

Winker, P., Fang, K.-T., 1997. Application of Threshold Accepting to the evaluation of the discrepancy of a set of points. SIAM Journal on Numerical Analysis 34, 2028–2042.