Efficient Optimization of Reinsurance Contracts using Discretized PBIL[1]

Omar Andres Carmona Cortes*, Andrew Rau-Chaplin†, Duane Wilson†, Ian Cook‡ and Jürgen Gaiser-Porter‡
*Informatics Academic Department
Instituto Federal de Educação, Ciência e Tecnologia do Maranhão
omar@ifma.edu.br, ocortes@dal.ca
†Risk Analytics Lab
Dalhousie University
arc@cs.dal.ca, dwilson@gmail.com
‡Global Analytics
Willis Group
cookiz@willis.com, gaiserporterj@willis.com

***Abstract*—Risk hedging strategies are at the heart of financial risk management. As with many financial institutions, insurance companies try to hedge their risk against potentially large losses, such as those associated with natural catastrophes. Much of this hedging is facilitated by engaging in risk transfer contracts with the global reinsurance market. Devising an effective hedging strategy depends on careful data analysis and optimization. In this paper, we study from the perspective of an insurance company a Reinsurance Contract Optimization problem in which we are given a reinsurance contract consisting of a fixed number of contractual layers and a simulated set of expected loss distributions (one per layer), plus a model of reinsurance market costs. Our task is to identify optimal combinations of placements such that for a given expected return the associated risk value is minimized. The solution to this high-dimensional multi-objective data analysis and optimization problem is a Pareto frontier that quantifies the best available trade-offs between expected risk and returns. Our approach to this reinsurance contract optimization problem is to adapt the evolutionary heuristic search method called Population Based Incremental Learning, or PBIL, to work with discretized solution spaces. Our multi-threaded Discretized PBIL method (or DiPBIL) is able to solve larger "real world" problem instances than previous methods. For example, problems with a 5% discretization and 7 or less contractual layers can be solved in less than 1h:20m, while previously infeasible problems that would have taken weeks or even months to run with as many as 15 layers can be solved in less than a day.**

***Keywords*—Financial Risk Management; Reinsurance Contract Optimization; Population Based Incremental Learning; Insurance and Reinsurance Analytics.**

I. Introduction

Risk hedging strategies are at the heart of financial risk management. As with many financial institutions, insurance companies try to hedge their risk against potentially large losses, such as those associated with natural catastrophes. Much of this hedging is facilitated by the global reinsurance market [1] (See Figure 1). Natural catastrophe reinsurers insure other insurance companies against the massive claims that can occur after events such as earthquakes, hurricanes, and floods. This transfer of risk is done in a manner similar to how a consumer cedes part of the risk associated with their private holdings by buying an insurance contract. This contract is defined in terms of a layer consisting of 1) a limit (i.e., the maximum payout), 2) a deductible or attachment (i.e., minimum loss triggering a claim), and 3) the share or placement (i.e., the percentage of losses in that layer that will be covered). Unlike the case of the consumer, the insurer has the ability to define complex multi-layered contracts and offer them to the reinsurance market. Doing so, it must carefully analyze the data it has on expected annual loss distributions and market reinsurance costs in order to identify contractual terms that maximize its expected reinsurance recoveries for each of a given set of risk tolerance values. In the insurance setting, typical risk measures include variance, Value at Risk (VaR) or a Tail-Value at Risk (TVaR) [1].

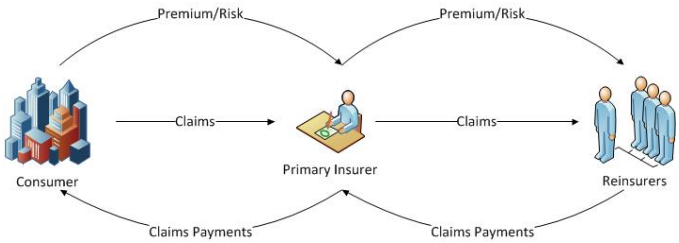


Figure 1. Risk and premium flows between consumers, primary insurers, and reinsurers

In this paper, we study from the perspective of an insurance company a *Reinsurance Contract Optimization problem*. Given a reinsurance contract consisting of a fixed number of layers and a simulated set of expected loss distributions (one per layer), plus a model of reinsurance costs, identifying optimal combinations of placements such that for a given expected return the associated risk value is minimized. The solution to this high-dimensional multi-objective optimization problem is

[1]This research was financially supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and Flagstone Re, Halifax, Canada under the Collaborative Research and Development Grant CRDPJ 412889-11. This project is also supported by the Science without Border program of CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico - Brazil) and Instituto Federal de Educação, Ciência e Tecnologia do Maranhão

a Pareto frontier that quantifies the available trade-offs between expected risk and returns, as illustrated in Figure 2.

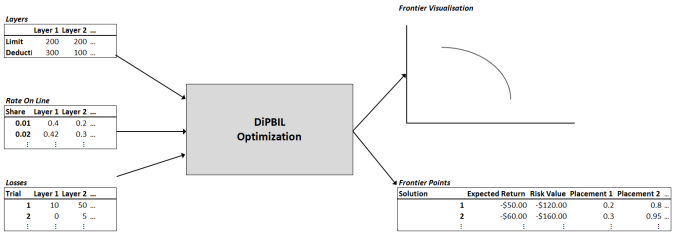


Figure 2. The studied reinsurance contract optimization problem: Inputs and Outputs

There are many heuristics methods that can be applied to optimization problems like this, such as Particle Swarm Optimization (PSO) [2], Differential Evolution (DE) [3], [4], Genetic Algorithms (GA) [5], Evolution Strategies (ES) [6] and Population-Based Incremental Learning (PBIL) [7]. Among these meta-heuristics we tried to use GA, nevertheless its performance was poor because demanded a lot of time to compute solutions considering our data set. Our approach is based on an adaptation of the evolutionary heuristic search method called Population Based Incremental Learning, or PBIL, which is a type of genetic algorithm where the genotype of an entire population is evolved rather than individual members and offers the following advantages. The algorithm is simpler than many standard genetic algorithms, works well in high dimensional spaces, and typically leads to equivalent or better results than standard GAs. Further, the algorithm was amenable to an important adaptation required by our problem, namely that solutions (i.e., placement values) must be discretized, typically in units of 10%, 5% or 1%, rather than being allowed to take on continuous values. Furthermore, PBIL also demands less computational power than the other methods since it is not based on genetic operators like crossover and selection.

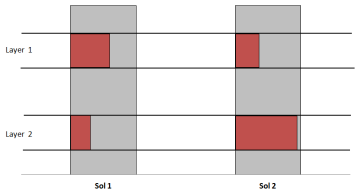
In the remainder of this paper, we first formally define our reinsurance contract optimization problem in Section 2. Then we describe a discretized PBIL method and how it can be applied to our problem in Section 3. In Section 4, we present a detailed performance analysis comparing our results to an enumeration method (both implemented in R) in terms of both speed and quality of result. Finally, we present the conclusions and future works in Section 5.

II. The Reinsurance Contract Optimization Problem

A. Reinsurance Business Basics

Insurance organizations, with the help of the global reinsurance market, look to hedge their risk against potentially large claims, or losses[1]. This transfer of risk is done in a manner similar to how a consumer cedes part of the risk associated with their private holdings.

Unlike the case of the consumer, whom is usually given options as to the type of insurance structures to choose from, the insurer has the ability to set its own structures and offers them to the reinsurance market. Involved in this process are decisions around the what type and the magnitude of financial



Solution	Layer	Limit	Deductible	Placement	Max Recovery	Premium
1	Layer 1	200	300	60%	120	$40
	Layer 2	200	100	20%	40	$40
2	Layer 1	200	300	30%	60	$25
	Layer 2	200	100	95%	190	$100

Figure 3. An example two layer reinsurance contract optimization problem with two sample solutions

structures, such as deductibles and limits, as well as the amount of risk the insurer wishes to maintain. The deductible describes the amount of loss that the insurer must incur before being able to claim a loss to the reinsurance contract, the limit describes the maximum amount in excess of the deductible that is claimable and the placement describes the percentage of the claimed loss that will be covered by the reinsurer.

Typically, companies try to hedge their risk placing multiple layers at once as illustrated in Figure 3. That is, they may have multiple sets of limit and deductible combinations. These different layers may also have differing placement amounts associated with them. At the same time, insurers are price takers in terms of the compensation paid to reinsurers for assuming risk. This compensation, or premium, depends on both the amount of risk associated with a layer and the placement amount of the layer. For this reason, it is important for insurers to choose placements when seeking to buy multiple layers. This optimal placement ensures that the insurer is able to maximize their returns on reinsurance contracts for potentially large future events.

In the remainder, we explore the use of optimization methods for finding optimal combinations of placements for multi-layer contracts from the prospective of a property-causality insurer. To simplify the problem description we focus on the primary contractual terms. Secondary terms such as the contractual costs associated with brokerage fee and contractual expenses, as well as provisions such as reinstatement premiums, are straightforward to add. As is typically done in reinsurance markets, contracts are assumed to be enforced for a one year period.

B. Reinsurance Costs

The basic cost of reinsurance to an insurer comes in the form of premium payments. As mention previously, the amount of premium paid for a layer can vary with the amount of the layer being placed in the market. In general, premiums are stated per unit of limit, also known as a *rate on line*. The cost of the reinsurance layer can then be expressed as follows:

$$p = \pi\mu(\pi, l, d) \times l \tag{1}$$

where p is the monetary value of the premium, μ is the rate on line, π is the placement, d is the deductible and l is the limit. For contracts with multiple layers, **(1)** can be generalized such that:

$$p = \vec{\mu}^T \mathbf{L} \vec{\pi} \tag{2}$$

where $\mathbf{L}$ is an $n \times n$ diagonal matrix of limits, $\vec{\mu}$ is a $n \times 1$ vector of rate on lines, $\vec{\pi}$ is a $n \times 1$ vector of placements, and n is the number of layers being placed. This matrix defines a model of expected reinsurance costs.

C. Reinsurance Recoveries

Losses affecting an insurer can be defined as a random variable X, such that:

$$X \sim f_X(x) \tag{3}$$

and f_X is some distribution that represents severity of X. These losses, once claimed, are subject to the financial terms associated with the contract they are being claimed against. Any one instance of X, x_i, then results in a claim of:

$$c_i = max\{0, min\{l, x_i - d\}\}\pi \tag{4}$$

where c_i is the value of the claim for i^{th} instance of X. Equation 4 can then be extended to contracts with multiple layers as follows:

$$c_i = \sum_{j=1}^{n} max\{0, min\{l_j, x_i - d_j\}\}\pi_j \tag{5}$$

where l_j, d_j and π_j are the limit, deductible and placement of the j^{th} layer respectively. In addition to this, many contracts allow for multiple claims in any given contractual year. The yearly contractual loss is then, assuming no financial terms that impose a maximum amount claimable, simply the sum of all individual claims in a given contractual year.

$$y_j = \sum_{i=1}^{n} c_{ij} \tag{6}$$

where y_j is the annual amount claimed for the j^{th} layer. The annual return for reinsurance contract is then defined as:

$$\begin{aligned} r &= \vec{y}^T \vec{\pi} - p \\ &= \vec{y}^T \vec{\pi} - \vec{\mu}^T \mathbf{L} \vec{\pi} \\ &= (\vec{y}^T - \vec{\mu}^T \mathbf{L}) \vec{\pi} \end{aligned} \tag{7}$$

where $\vec{y}$ is an $n \times 1$ vector of annual claims for each layer.

D. The Risk Value and Optimization Problem

Given a fixed number of layers and loss distributions the insurer is then faced with selecting an optimal combination of placements. As with most financial structures, the problem faced is selecting an optimal proportion, or placement, of each layer such for a given expected return on the contracts the associated risk is minimized. This is generally done by using a risk value such as a variance, Value at Risk (VaR) or a Tail-Value at risk (TVaR). The Tail-Value at Risk is also referred to as a conditional Value at Risk (CVaR) or a conditional tail expectation (CTE). Unlike, the traditional finance portfolio problem, in the insurance context a claim made, or loss, to the contract is income to the buyer of contract. This means, from the perspective of the insurer, they wish to maximize the amount claimable for a given risk value. In doing so they minimize amount of loss the insurer may face in a year.

Equation 7 can then be rewritten in matrix format such that:

$$\mathbf{R} = (\mathbf{Y} - \mathbf{ML})\vec{\pi} \tag{8}$$

where $\mathbf{R}$ is a $m \times 1$ vector of recoveries, $\mathbf{Y}$ is a $m \times n$ matrix of annual claims and $\mathbf{M}$ is a $m \times n$ matrix of rates on line. Since the same year is being simulated each row in matrix $\mathbf{M}$ is the same. This formulation leads to a optimization problem as follows:

$$\begin{aligned} &maximize && VaR_\alpha(\mathbf{R}(\pi)) \\ &s.t. && E(\mathbf{R}(\pi)) = a \end{aligned} \tag{9}$$

Given that the expected return a is not specified **(9)** can be rewritten to a Pareto Frontier problem, such that:

$$maximize \qquad VaR_\alpha(\mathbf{R}(\pi)) - qE(\mathbf{R}(\pi)) \tag{10}$$

where q is a risk tolerance factor greater than zero.

This problem can be approached in using numerous methods. Mistry et al. [14] use an enumeration approach by discretizing the search space for the each layer's placements. The discretization of the placements may be desirable for practical reasons (i.e., a placement with more than two decimal places may be invalid in negotiations) and the full enumeration method lends itself well to parallel computation. However, the computational time to evaluate all possible combinations increases exponentially as the number of layers and the resolution of the discretization increases. This renders the enumeration approach infeasible for many practically sized problems.

Mitschele et al. introduced the use of heuristic methods for addressing reinsurance optimization problems [13]. They show the power of two multi-objective evolutionary algorithms in finding non-dominated combinations, in comparison to the true non-dominated set of points. Mitschele et al., however, work exclusively in continuous space and focus on algorithms that change the limit and deductible aspects of a reinsurance contract, so there methods are not directly applicable here.

III. Discretized Population Based Incremental Learning

Population-Based Incremental Learning (PBIL) was first proposed by Baluja [7]. The algorithm's populations are encoded using binary vectors and an associated probability vector, which was then updated based on the best members of a population. Unlike other evolutionary algorithms, which transform the current population into new populations, a new population is generated at random using the updated probability vector on each generation. Baluja describes his algorithm as

a "combination of evolutionary optimization and hill-climbing" [7].

Since Baluja's work, extensions to the algorithm have been proposed for continuous and base-n represented search spaces [8], [9], [10]. The extension to continuous search spaces using histograms (PBIL$_H$) and real-code (RPBIL) suggest splitting the search space into intervals, each with their own probability [8], [10]. For multivariate cases, the probability vector is then substituted for a probability matrix, in such that each row or column of the matrix represents a probability vector for any given independent variable.

Algorithm 1 describes the discretized PBIL (DiPBIL) method used in this paper in terms of the following tunable parameters:

- N_G = Total number of generations or iterations
- n = Size of each population
- I = Number of Increments (i.e., the discretization)
- LR_2 = Learning Rate in base 2
- NLR_2 = Negative Learning Rate in base 2
- M_R = Mutation Rate
- M_S = Mutation Shift
- q = Number of best results to be used in updating

Algorithm 1: DiPBIL

Input: N_G, I, LR_2, NLR_2, M_R, M_S, n, q, function name(*fun*)
Output: $\mathbf{x}_G^{best}$, f_G^{best}
Initialization: P_{ij} = 1/I , LR_N, NLR_N, $\mathbf{x}_G^{best}$={}, $\mathbf{x}_i^{best}$={}

for i = *1* to N_G **do**
  Generate a population **X** of size n from P_{ij} ;
  Evaluate **f** = fun(**X**);
  Find $\mathbf{x}_G^{best}$ from the current and previous populations;
  Find $\mathbf{x}_i^{best}$ for top q-1 members of the current population;
  Update P_{ij} based on $\mathbf{x}_G^{best} \cup \mathbf{x}_i^{best}$ using LR_N and NLR_N;
end

While PBIL$_H$ and RPBIL support continuous search spaces, a similar method can be applied to a discretized search space. Here, we substitute the intervals for equidistant increments in the lower and upper bounds of the search space. This can also be related to a base-n representation, where each point in a given probability vector represents one number [9]. In the same spirit as PBIL$_H$ and RPBIL the probability matrix is initialized with all increments having equal probability and is updated after every generation with the best combinations member (see Algorithm 1). The updating of each vector in the matrix, however, is done using the base-n method, with an adjusted learning rate and updating function [9]. RPBIL suggests its own updating function which exponentially increases the probabilities as you move toward intervals that are closer to the best individuals [8]. The base-n updating method, however, has the side effect of slightly increasing the probability of increments further from the best individuals in the search space and may allow for a chance at more population diversity [9]. To ensure more population diversity from across generations, the probability matrix is updated with best member from previous generations as well as the top q members from the current generation. This modifies the updating equation slightly, such that:

$$p_{ij}^{NEW} = \sum_{k=1}^{q} p_{ij}^{OLD} \frac{LF_{ijk}}{q} \tag{11}$$

where LF_{ijk} is the i^{th} learning factor, as described in [9], for the k^{th} best result for the j^{th} variable.

IV. Performance Evaluation

This section presents the experimental evaluation of our reinsurance contract optimization technique. We first discuss our setup and methodology as well as the data sets used for the evaluation. We then present the performance results obtained in terms of quality of solutions and performance.

A. Experimental Setup and Methodology

We have implemented our discretized PBIL method using R and RStudio [11] and the doParallel parallelization package [12]. Our experimental platform consisted of a SunBlade server x6440, with four Quad-core AMD Opteron 8384 (2.7GHz) processors and 32 GB Ram, running Red Hat Enterprise Linux 4.8. The prototype code was written in R version x64 2.15.0 and doParallel version 1.0.1 with socket based connections.

All single threaded times were measured as wall clock times in seconds. All mutli-threaded times were measured as the wall clock time between the start of the first process and the termination of the last process. We will refer to the latter as *parallel wall clock time*.

For our experiments, we used anonymized industry data consisting of between 7 and 15 layers, with loss distributions represented by 50,000 trial loss sets, and a rate on line or reinsurance cost model defined in 10% increments with linear interpolation used between data points.

In all runs of DiPBIL, the following values were used for the fixed parameters: $LR_2 = 0.1$, $NLR_2 = 0.01$, $M_R = 0.02$, $M_S = 0.05$, and $q = 3$. These parameter settings were chosen based on recommendations from the literature followed by empirical testing. In experiments that varied the number of iterations, discretization, and/or population size the following values were used: $N_G = 500$, 1000 or 2000; $I = 10\%$ or 5%; and ($n = 100$, 200 or 400).

Our experiments proceeded in the following steps.

- Comparison of DiPBIL vs exact methods: Quality of results.
- Comparison of DiPBIL vs exact methods: Speed.
- Performance of multi-threaded DiPBIL.
- Pushing the envelope.

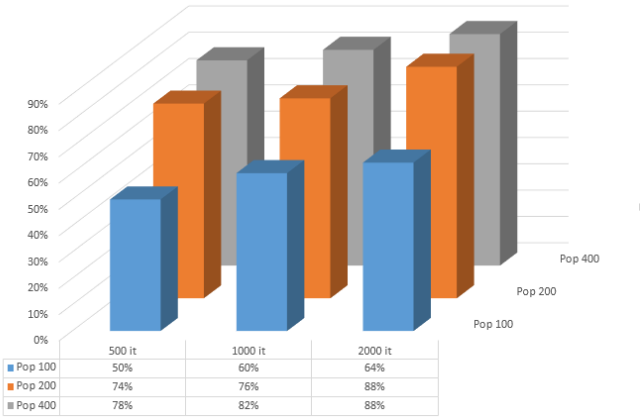


Figure 4. Percentage of time DiPBIL finds the same solution as the exact method for varying iterations and population size

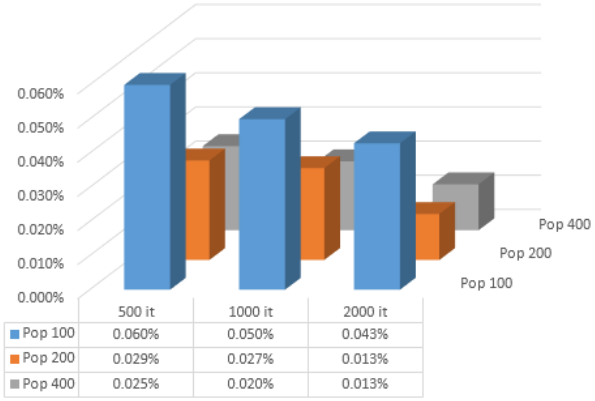


Figure 5. Average error when DiPBIL does not find the same solution as the exact method for varying iterations and population size

B. Comparison of DiPBIL vs exact methods: Quality of results.

Figure 4 shows the percentage of time DiPBIL finds exactly same solution as the exhaustive enumeration approach for varying iterations and population size on a problem with 7 layers and a 5% discretization. Each data point represents the average of 50 experiments. As expected, increasing iterations and population size increases the probability of finding exactly the same solution as the exhaustive enumeration approach, with $N_G = 2000$ iterations on a population size of $n = 200$ providing the best quality vs. time trade-off.

Figure 5 shows the average error for the cases where the exact solution is not found. We observe that the average error is always less than our predetermined error tolerance of 1%. The largest average error occurs, as expected, with the smallest population size and iteration count, but even in this case the average error is only 0.06% and with 2000 iterations on a population size of 200 the average error drops to just 0.013%. On other words, with 2000 iterations and population size of 200 the difference between the average of 50 runs and the optimum given by the enumeration method is 0.013%.

C. Comparison of DiPBIL vs exact methods: Speed.

The main reason behind the use of PBIL is the infeasible time to run the enumeration method, especially in large problems with many layers. Figure 6 illustrates the required time for different level of discretizations and number of layers, where shares 01, 05, 10 and 25 depict the discretization levels of 1%, 5%, 10% and 25%, respectively. For example, considering 7 layers and 5% of discretization, the enumeration method takes more than a week to get the answer.

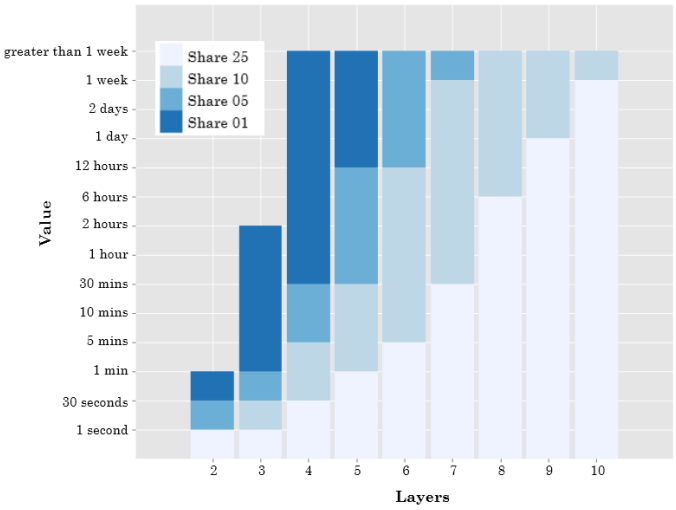


Figure 6. Required time for different number of layers and discretization

On the other hand, Figure 7 shows the time on a single core to compute a single point on efficient frontier for a 7 layer and 5% discretization problem. Note that the enumeration method takes over 6 days to run with a 5% discretization. The DiPBIL based approaches take between 100 and 1100 seconds to run depending on the population size and number of iterations. It is also important to note that DiPBIL is relatively insensitive to the discretization level, running in roughly the same time at 1% discretization, while reducing the discretization to 1% for the enumeration method will (by our estimation) increase the computing time to over a year. While this graph captures the relative performance of the methods all of the reported values based on a prototype R implementation could likely be reduce by a significant factor with a optimization implementation in C/C++.

D. Performance of multi-threaded DiPBIL.

Figure 8 shows the time and Figure 9 the corresponding speedup achieved by the multi-threaded version of DiPBIL, while computing a 32 point efficient frontier for a realistic 7 dimensional Treaty Optimization problem using a population size of 200 and 2000 iterations. Even with prototype code written in R a 75% speedup is achieved with results obtained in about 1h 20m. Early indications suggest that a tuned C/OpenMP implementation would be significantly faster.

E. Pushing the envelope.

In order to explore the limits of how large a treaty optimization problem we could effectively solve, we executed

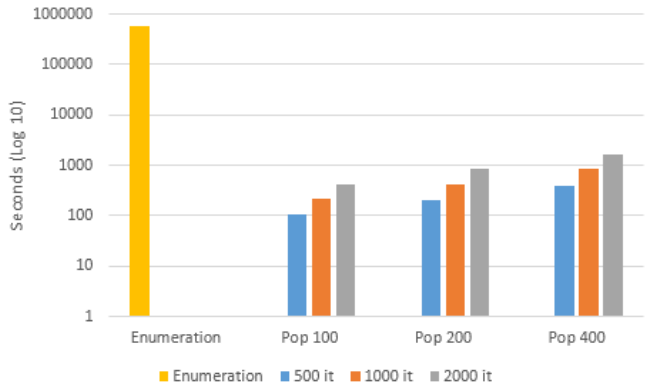


Figure 7. Comparing speed of DiPBIL for varying dimensionality

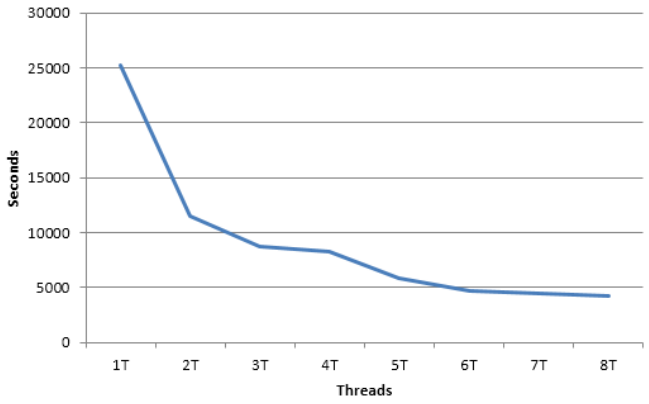


Figure 8. Performance of multi-threaded DiPBIL with increasing thread count

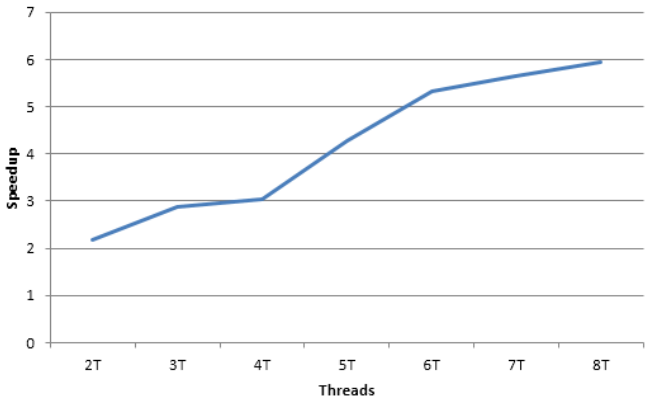


Figure 9. Speedup of multi-threaded DiPBIL with increasing thread count

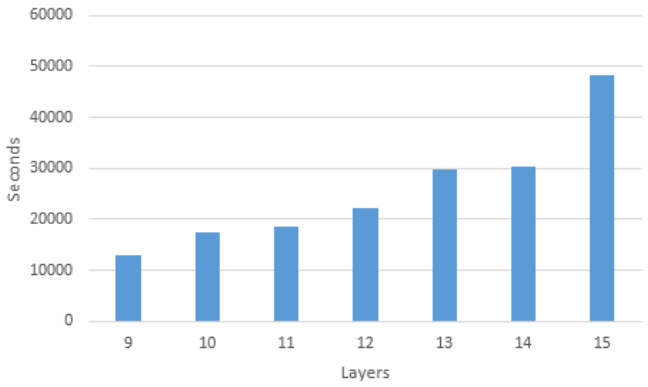


Figure 10. Time for multi-threaded DiPBIL to solve high dimensional treaty optimization problems with at fine level of discretization (5%)

multi-threaded DiPBIL on problems with between 9 and 15 dimensions and a 5% discretization with and measured the wall clock time. Figure 10 shows the results where each data point represents the average time to compute a 32 point frontier. We observed that DiPBIL was able to solve larger instances of the treaty optimization problem than in any previously reported implementation. Treaty optimization problems with as many as 9 layers were solvable in under 4 hours and massive problems involving 15 layers took significantly less than a day. This is a significant improvement since the run time for algorithms solving the treaty optimization problem typically grow exponentially in the number of layers.

V. Conclusion and Future Work

In this paper, we have studied a reinsurance contract optimization problem and shown that an approached based on a discretized adaptation of Population Based Incremental Learning generates high quality solutions in a time efficient manner. Either, Our multi-threaded DiPBIL method is able to solve "real world" problem instances with a 5% discretization and 7 or less layers in less than 1h:20m, and with up to 15 layers in less than a day. In contrast to exact enumeration methods, only treaty optimization problems with less than 7 layers are solvable in a day, while problems with 7 or more layers require a week or more of computation, or are simply infeasible.

This approach makes solving "real-world" problems with more than 8 layers and a 5% discretization feasible in a way it previously was not. Moreover, the differences between the DiPBIL and the enumeration method is not significant (lower than 0.06%) even when the lowest configuration (500 iterations and a population size equals to 200) is used.

In the future, we plan to extend our analysis to Differential Evolution and Particle Swarm Optimization. Also basic approach by examining alternative heuristic search strategies that might also support the required solution space discretization, evaluate the performance gains achievable with an optimized C/OpenMP implementation, and add constraints necessary to support secondary financial terms. Moreover, a multi-objective approach of DiPBIL is also in development.

References

[1] Cai, J., Tan, K. N., Weng, C. and Zhang, Y., Optimal reinsurance under VaR and CTE risk measures. *Insurance: Mathematics and Economics*, 43, 185-196, 2007.

[2] Kennedy, J. and Eberhart, R., Particle swarm optimization, Proc. of IEEE International Conference on Neural Networks , vol.4, pp. 1942-1948, 1995.

[3] Storn, R. and Price, K., Differential Evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces, Technical Report TR-95-012, ftp.ICSI.Berkeley.edu/pub/techreports/1995/tr-95-012.ps.Z, 1995.

[4] Storn, R. and Price, K., Minimizing the real functions of the ICEC96 contest by differential evolution, Proc. of IEEE International Conference on Evolutionary Computation, Nagoya, Japan, 1996.

[5] Michalewicz, Z., Genetic Algorithms + Data Structure = Evolution Programs, 3 ed, Springer, 1996.

[6] Yao, X., Liu, Y. and Lin, G., Evolutionary programming made faster, *IEEE Transactions on Evolutionary Computation*, v.3, n.2, pp. 82-102, 1999.

[7] Baluja, S., Population based incremental learning. *Technical Report*, Carnegie Mellon University, 1994

[8] Bureerat, S., Improved population-based incremental learning in continuous spaces. *Soft Computing in Industrial Applications*, 96, pp. 77-86, 2011

[9] Servais, M.P., Jager, G. and Greene, J.R., Function optimisation using multi-base population based incremental learning. *PRASA, Rhodes University*, 1997

[10] Yuan, B. and Gallagher, M., Playing in continuous spaces: Some analysis and extension of population-based incremental learning. *CEC2003, CA, USA*, pp. 443-450, 2003.

[11] http://www.rstudio.com/, Last Visit 20-May-2013.

[12] http://cran.r-project.org/web/packages/doParallel/index.html, Last Visit 20-May-2013.

[13] Mitschele, A., Oesterreicher, I. and Schlottmann, F. and Seese, D., Heuristic optimization of reinsurance programs and implications for reinsurance buyers. Operations Research Proceedings, pp. 287-292, 2006.

[14] Mistry, S. (n.d.), Gaiser-Porter, J. and McSharry, P. and Armour, T., *Parallel Computation of Reinsurance Models*. Unpublished Manuscript.

[15] http://www.ace-net.ca/wiki/Glooscap, Last Visit 20-May-2013.