A Modified Multi-objective Differential Evolution Algorithm with Application in Reinsurance Analytics

Omar Andres Carmona Cortes
Instituto Federal do Maranhão
Informatics Department
São Luis, MA, Brazil
Email: omar@ifma.edu.br

Andrew Rau-Chaplin
Dalhousie University
Faculty of Computer Science
Halifax, NS, Canada
Email: arc@cs.dal.ca

Abstract—**In the reinsurance marketplace, the risk of financial loss in the event of natural catastrophes (such as earthquakes, hurricanes and floods) is exchanged between market participants for a premium. Here, prudent risk management takes the form of a hedge against the risk of a contingent uncertain loss in exchange for a payment. Reinsurance contracts that define the terms of the transfer are elaborated multi-layered financial treaties that represent complex trade-offs between expected return and risk. Formulating an effective risk transfer strategy depends on a careful multi-objective optimization process. In this paper, we study from the perspective of an insurance company the Reinsurance Contract Optimization problem in which, given the structure of a multi-layered reinsurance contract, we are required to discover specific contractual terms that capture the best trade-offs between expected return and risk for the insurer. Our approach is based on an adaptation of Multi-Objective Differential Evolution. In searching for the best mutation operators, we performed an experimental analysis on large-scale real problem instances using industrial datasets and evaluated five different mutation operators. Our experimental results indicate that those mutation operators based on selecting non-dominated individuals from the archive tend to produce better outcomes. Since speed is critical in this application, we also developed a parallel version achieving a speedup up to 9.3 on a 16 core machine.**

Keywords–Risk Analytics; Differential Evolution; Multi-objective; Parallel Computing.

I. Introduction

Many real world applications involve the optimization of two or more conflicting objectives, where the search for solutions generates a Pareto frontier [1], on which no particular solution is better than another. Solutions on the Pareto frontier represent trade-offs between multiple objectives. Given these trade-offs, human experts can select final solutions based on other, often more qualitative, criteria.

An interesting real world application in computational finance is the Reinsurance Contract Optimization (RCO) problem. RCO is a treaty optimization problem in which we are given as input the structure of a complex risk transfer treaty consisting of a fixed number of contractual layers, a simulated set of expected loss distributions (one per layer), and a model of reinsurance market costs [2][3]. The task is to generate as output the best set of shares, a key financial parameter, which balance the expected return and the risk. In this context, typical risk measures include variance, Value at Risk (VaR) or a Tail-Value at Risk (TVaR) [4].

An enumeration method can be used for solving the RCO problem; however this approach presents two main problems: 1) it has to be discretized, demanding some changes in numerical algorithms and 2) in practice, it is only applicable to small problems instances (i.e., 2 to 4 layers), whereas real instances of the RCO problem can have 7 or more layers. For instance, a 7 layered problem can take several week to be solved with a 5% level of discretization on the search space using the enumeration method as presented in [2]. As a consequence, it is important to explore alternative methods for addressing this type of problem.

In this context, evolutionary algorithms, such as differential evolution (DE) [5], seem to be a natural choice. DE is reasonably simple to implement and has been successfully used in many applications including Reservoir System Optimization [6], Communication Systems [7], and Speaker Recognition [8]. Risk and reinsurance problems have also been tackled using evolutionary methods, such as in [9][10][11]; however, the focus in these particular applications was on stop loss and ruin prediction, *i.e.*, a very different problem than the RCO problem studied in this paper. In the context of RCO, the first studies using evolutionary methods were [2] and [3].

While the techniques proposed in these papers performed significantly better than the enumeration approach, they suffered from a critical drawback. They were based on single-objective optimization methods in which the risk could be optimized only for a given expected return value in any one call to the optimizer. Consequently, creating a Pareto frontier that covers a range of expected return values was very time-consuming, making it unsuitable for many industrial scale problems. In [12], a faster vector evaluated differential evolution method for RCO was presented. While this approach was multi-objective, producing the whole Pareto frontier at once, it suffered from solution quality issues in that it often produced Pareto frontiers with holes or large gaps between solutions especially in the critical middle portion of the curve.

In this paper, we present a modified Differential Evolution for Multi-objective Optimization (DEMO) [13] algorithm - simpler than Multi-Objective Differential Evolution Algorithm (MODEA) [14] - which is both fast and solves the previously noted gaps problem, thus producing high quality Pareto frontiers. Our approach uses an archive of previously identified solutions in order to avoid losing non-dominated candidates as the optimization converges. Solutions were lost when the number of non-dominated solutions is truncated by crowding distance when it is bigger than the population size in the original version. Additionally, we present a study of different mutation operators, and propose the use of a non-dominated solution in the mutation step, instead of any random individual. An experimental evaluation of our modified DEMO method applied to the RCO problem demonstrates that it can solve extremely large real-world RCO problems with between 7 and 15 layers (subcontracts) in under three minutes. Our experimental results indicate that the quality of solutions, when evaluated in terms of the average size and hyper volume of the generated Pareto frontiers, is high and the previously noted gaps problem, especially in the critical middle portion of the curve, is now largely absent.

The remainder of this paper is structured as follows: Section II presents fundamentals concepts of multi-objective problems and an introduction to the RCO problem; Section III shows how DEMO works and our proposal; Section IV introduces the metrics that were applied and the results, including some parallelization features; finally, Section V presents conclusion and future work.

II. Multiobjective Problems

A Multi-objective Optimization Problem (MOP) has to address two or more conflicting objective function [15] at the same time. The resulting solution is a Pareto frontier, *i.e.*, a set of points where no solution is better than another one. Otherwise, the global optima would be only one point in the search space [12]. Thus, assuming that a solution to a MOP is a vector in a search space X with m elements. A function $f : X \to Y$ evaluates the quality of a solutions mapping it into an objective space. Therefore, a multi-objective problem is defined as presented in (1), where f is a vector of objective functions, m is the dimension of the problem and n represents the number of objective functions.

$$Max\ y = f(x) = (f_1(x_1, ..., x_m), ..., f_n(x_1, ..., x_m)) \quad (1)$$

In order to determine whether a solution belongs to the Pareto frontier or not, it is necessary to use the concept of optimality (i.e., Pareto dominance), which states that given two vectors x, $x* \in \Re$ and $x \neq x*$, x dominates $x*$ (denoted by $x \succeq x*$) if $f_i(x)$ is not worse than $f_i(x*), \forall i$ and there exist at least one i where $f_i(x) > f_i(x*)$ in maximization cases and $f_i(x) < f_i(x*)$ otherwise. Hence, a solution x is said Pareto optimal if there is no solution that dominates x, in such case, x is called non-dominated solution. Mathematically, assuming a set of non-dominated solutions $\wp$, a Pareto frontier(pf) is represented as $pf = \{f_i(x) \in \mathbb{R} | x \in \wp\}$.

A. A Treaty Optimization Problem: RCO

Insurance organizations, with the help of the global reinsurance market, look to hedge their risk against potentially large claims, or losses [4]. This transfer of risk is done in a manner similar to how a consumer cedes part of the risk associated with their private holdings [2]. The claims received by the insurer in case of a natural catastrophe are also referred to as expected return.

The reinsurance contract optimization consists 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 [2]. Hence, the main task is to discover the best combination of shares, also known as placements, which leads to a set of trade-offs between expected return and risk. In other words, insurance companies aim to hedge their risk against potentially large claims, or losses [4], especially those ones resulting from natural catastrophes. When these trade-offs are set, the insurance companies are able to offer them to the reinsurance market.

Overall, the purpose is both to maximize the amount of return ($) received from the reinsurance company and maximize the risk transferred to it. Doing so, the insurance companies minimize the loss faced per year in case of a natural disaster. In this context, (2) represents the problem in terms of optimization, where VaR is a risk metric, $\mathbf{R}$ is a function in term of placements (π) and E is the Expected Value. In probability theory, the expected value, usually denoted by E[X], refers to the value of a random variable X that we would "expect" to find out if we could repeat the random variable process an infinite number of times and take the average of the values obtained. For further details about the problems, refer to [2] and [4].

$$\begin{array}{ll} maximize & f_1(x) = VaR_\alpha(\mathbf{R}(\pi)) \\ maximize & f_2(x) = E[\mathbf{R}(\pi)] \end{array} \quad (2)$$

III. Differential Evolution Multi-Objective (DEMO)

The DEMO algorithm is shown in Figure 1, where we can observe that it is similar to the canonical version of DE whose strategy is DE/Rand/1 [16]. The differences start in line 16 when the new population is selected for the next iteration. Thus, if a new individual ($indiv$) dominates the target one (Pop_i) then the new one is added into a new population; if the target individual dominates the new one then the target element is added into the new population; otherwise, both individuals go to the new population. The dominance process builds a new population whose size rages from pop_size to $2 \times pop\_size$. Finally, if the size of the new population is larger than pop_size then the new individuals which go to next iteration are selected by crowding distance ($select\_cdistance$ function).

A. Our Proposal

The main drawback of the original DEMO was not maintaining an archive, thereby loosing good solutions when the number of non-dominated points overcomes the size of the population. Taking this into account, we changed the original algorithm in two parts. Firstly, we introduce an archive in the algorithm (after line 31), which is maintained on each iteration in order to do not lose non-dominated solutions from one iteration to another due to the crowding distance algorithm in line 30. Secondly, we tested some mutation operators (line 6 in the Figure 1) as presented in (4), (5), (6), and (7). Unless the original mutation operator which uses three any random individuals in order to build the F vector, in (4), we uses a random individual from the set of non-dominated ones. Thus, it is necessary to compute the non-dominated set between lines 3 and 4, *i.e.*, before starting the loop which deals with the population. (5) is similar to the previous one; however, F is a random number between 0 and 1. Then, in (6) and (7), we

```
$Pop \leftarrow$ generate_pop$(n, d)$
$fit \leftarrow$ evaluate$(Pop)$
while (Stop Criteria is FALSE) do
    for $i$ = 1 to #pop_size do
        $idx \leftarrow$ select_indiv(3)
        $v \leftarrow Pop_{idx_3} + F * (Pop_{idx_1} - Pop_{idx_2})$
        for $j$ = 1 to dimension do
            nj = rand()
            if ($nj < CR$) then
                indiv$\leftarrow v_j$
            else
                indiv$\leftarrow Pop_i j$
            end
        end
        $fit' \leftarrow$ evaluate$(indiv)$
        if ($fit'$ dominates $fit_i$) then
            $pop' \leftarrow indiv$
            $nf \leftarrow fit'$
        else if ($fit_i$ dominates $fit'$) then
            $pop' \leftarrow Pop_i$
            $nf \leftarrow fit_i$
        else
            add $indiv$ and $Pop_i$ into $pop'$
            add $fit$ and $fit'$ into $nf$
        end
    end
    if (size($pop'$) == size($Pop_i$))) then
        $Pop \leftarrow pop'$
    else
        $[Pop, fit] \leftarrow$ select_cdistance$(pop', nf)$
    end
end
```

Figure 1. DEMO Algorithm

randomly pick up the first individual from the archive which currently contains the best solutions found by the algorithms. The difference between the last two equations is the use of F which is randomly chosen in (7).

$$v \leftarrow non\_dom_{idx} + F * (Pop_{idx_1} - Pop_{idx_2}) \quad (3)$$
$$v \leftarrow non\_dom_{idx} + Rand() * (Pop_{idx_1} - Pop_{idx_2}) \quad (4)$$
$$v \leftarrow archive_{idx} + F * (Pop_{idx_1} - Pop_{idx_2}) \quad (5)$$
$$v \leftarrow archive_{idx} + Rand() * (Pop_{idx_1} - Pop_{idx_2}) \quad (6)$$
$$(7)$$

In the next section, the mutation operators will be referred to as M1 (canonical mutation), M2 (4), M3 (5), M4 (6), and M5 (7).

IV. Computational Experiments

All tests were conducted using R version 2.15.0 and RStudio on a Windows 7 64-bit Operating System running on an Intel i7 3.4 Ghz processor with 4 physical cores and hyper threading, with 16 GB of RAM. We executed the parallel version in an Intel Xeon comprising of two Xeon processors E5-2650 running at 2.0 Ghz with 8 cores and hyper threading and 256 GB of memory. The experiments used $F = 0.7$ or a random F, and $CR = 0.9$ considering 250, 500 and 1000 iterations with a population size equals to 50. Further, all averages are calculated in 30 trials. Our data set is composed by 7 layers of real anonymized data. The 15 layers data set was synthetically created based on the 7 layers one.

A. Metrics

In this section, we discuss the experimental evaluation of MODE algorithm. Firstly, the average number of non-dominated points (number of solutions) found in the Pareto frontier was determined. Secondly, the average hyper volume, which is the volume of the dominated portion of the objective space as presented in (8), was measured, where for each solution $i \in Q$ a hypercube v_i is constructed. The extreme points are those one belonging to the Pareto front. Having each v_i, we calculated the final hyper volume by the union of all v_i. The final number of solutions after all trials is showed as well.

$$hv = volume(\bigcup_{i=1}^{|Q|} v_i) \quad (8)$$

Thirdly, the dominance relationship between Pareto frontiers obtained with different mutation operators was calculated as depicted in (9). Roughly speaking, $C(A, B)$ is the percentage of the solutions in B that are dominated by at least 1 solution in A [17], therefore, if $C(A, B) = 1$ then all solutions in A dominate B, and $C(A, B) = 0$ means the opposite. It is important to notice that this metric is neither complementary by itself nor symmetric, *i.e*, $C(A, B) \neq 1 - C(B, A)$ and $C(A, B) \neq C(B, A)$ making important to compute it in both direction: $C(A, B)$ and $C(B, A)$. Finally, the resulting frontiers can be reviewed by experts for reasonability. For further details about the use of these metrics see [15].

$$C(A, B) = \frac{|\{b \in B | \exists a \in A : a \preceq b\}|}{|B|} \quad (9)$$

In terms of parallelism, we calculated the speedup according to $speedup = \frac{T_s}{T_p}$, where T_s is the execution time considering one thread and T_p represents the time in parallel using p threads. This kind of metric is called weak speedup and it was suggested in Alba [18] because the code is exactly the same regardless the number of threads, thus it is not necessary to guarantee that the serial version is the best one.

B. Results

Table I presents the average of number of solutions, the hypervolume, the elapsed time and the final number of solutions for 7 layers. As we can observe, using non-dominated individuals either from the population or the archive tend to produce more number of solutions; however, better hypervolumes are obtained by the version using the archive. Also, the difference in terms of time is not significant between mutation operators. Figure 2 shows the final Pareto frontier for 7 layers where we can noticed that visually it is not possible to identify which mutation operator is the best for this particular application.

Table II shows the coverage metric between the different mutation operators where it noticeable that the canonical mutation M1 dominates only between 5% and less than 1% of the other approaches. On the other hand, M5 dominates more

TABLE I. METRICS FOR 7 LAYERS AND 250 ITERATIONS

	#NS	Hypervolume	Time	Final #NS
M1	83.73	2.19E+15	105.18	226
Stdev	5.59	9.40E+13	3.40	
M2	151.30	1.96E+15	104.50	339
Stdev	12.91	3.52E+14	2.35	
M3	170.77	1.80E+15	103.41	330
Stdev	13.95	3.16E+14	2.60	
M4	203.67	2.23E+15	105.08	374
Stdev	10.80	2.60E+14	2.19	
M5	232.93	2.21E+15	105.63	383
Stdev	11.44	2.43E+14	2.13	

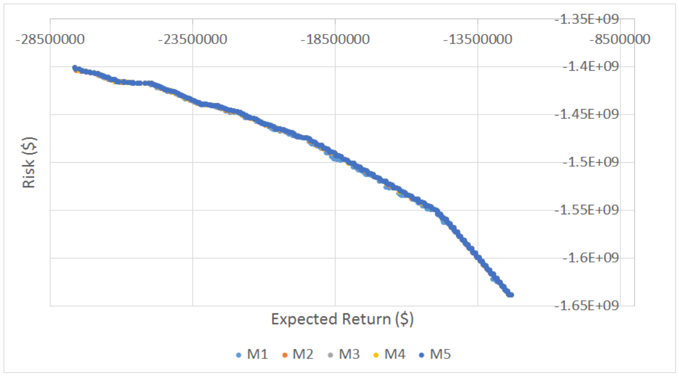

Figure 2. Final Pareto frontier for 7 layers and 250 iterations

solutions from the other operators. For example, M5 dominates 78% of solutions from M1 and 30% of solutions from M4. On the other hand, M4 dominates only 9% of M5 solutions, demonstrating that M5 is the most effective in this case.

TABLE II. COVERAGE FOR 7 LAYERS AND 250 ITERATIONS

	M1	M2	M3	M4	M5
M1	-	0.05	0.06	0.03	0.007
M2	0.66	-	0.19	0.098	0.034
M3	0.66	0.30	-	0.14	0.06
M4	0.73	0.39	0.28	-	0.09
M5	0.78	0.48	0.31	0.30	-

Table III illustrates the same metrics aforementioned for 250 iterations and 15 layers. In this case, the behavior is similar to the previous one in terms of number of solution and time; however, M4 presented the best hypervolume. Moreover, visually there are some differences between the performance of the operators as we can see in Figure 3, where M4 and M5 seem to be better than the other operators. The coverage metric in Table IV indicates that M5 dominates more solutions than M4, therefore, the bigger hypervolume might be caused by the non-dominated points in the beginning of the Pareto frontier curve.

The behavior for 7 and 15 layers using 500 iterations are presented in Tables V and VII. As we can see, the results are similar to the previous ones including the final Pareto frontier (Figures 4 and 5) and the coverage rates in Tables VI and VIII; but, now the differences are in a smaller scale. This result is expected because as we increase the number of iterations the differences tend to be smaller. On the other hand, this number of iterations is not sufficient for approximating the results using 15 layers because this last one is a much harder problem to solve.

TABLE III. METRICS FOR 15 LAYERS AND 250 ITERATIONS

	#NS	Hypervolume	Time	Final #NS
M1	55.23	2.96E+15	166.66	104
Stdev	4.70	2.87E+14	5.61	
M2	93.90	2.54E+15	165.20	220
Stdev	8.86	5.83E+14	5.81	
M3	139.90	2.39E+15	166.32	354
Stdev	15.20	6.38E+14	2.77	
M4	145.23	3.40E+15	166.43	340
Stdev	10.12	8.53E+14	4.41	
M5	188.67	3.05E+15	168.57	390
Stdev	12.71	6.81E+14	2.38	

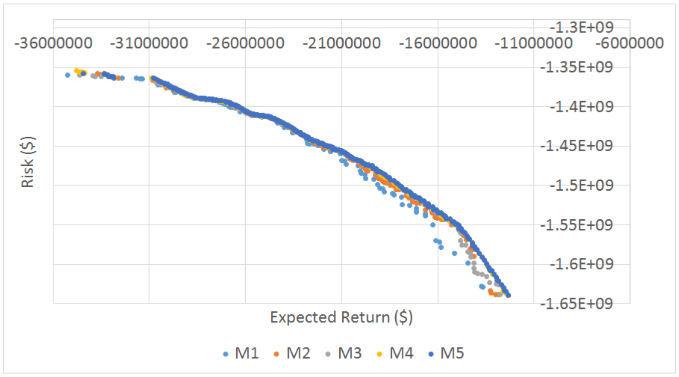

Figure 3. Final Pareto frontier for 15 layers and 250 iterations

TABLE IV. COVERAGE FOR 15 LAYERS AND 250 ITERATIONS

	M1	M2	M3	M4	M5
M1	-	0.036	0.025	0.006	0.000
M2	0.88	-	0.087	0.07	0.036
M3	0.89	0.69	-	0.30	0.10
M4	0.95	0.80	0.42	-	0.11
M5	0.98	0.89	0.70	0.61	-

TABLE V. METRICS FOR 7 LAYERS AND 500 ITERATIONS

	#NS	Hypervolume	Time	Final #NS
M1	101.40	2.26E+15	209.48	237
Stdev	9.19	5.40E+13	4.18	
M2	182.43	2.20E+15	208.09	373
Stdev	11.74	2.01E+14	4.96	
M3	191.57	1.65E+15	205.97	367
Stdev	16.82	4.68E+14	4.88	
M4	240.77	2.15E+15	206.66	399
Stdev	10.36	2.75E+14	4.87	
M5	290.17	2.20E+15	207.13	397
Stdev	12.02	2.33E+14	5.59	

TABLE VI. COVERAGE FOR 7 LAYERS AND 500 ITERATIONS

	M1	M2	M3	M4	M5
M1	-	0.083	0.087	0.01	0.002
M2	0.616	-	0.19	0.085	0.015
M3	0.60	0.22	-	0.102	0.025
M4	0.718	0.365	0.29	-	0.053
M5	0.747	0.437	0.34	0.235	-

The results for 1000 iterations for 7 and 15 layers are presented in Tables IX and XI. Visually, the results for 7 layers in Figure 6 are the same; however, Table X indicates that the differences are still there. Looking at the number of the Pareto frontier, we will see similar solutions between M1 and M5; nonetheless, the M5 solutions dominates the

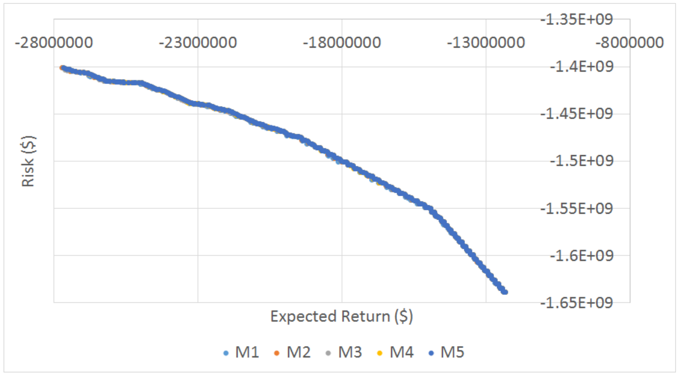

Figure 4. Final Pareto frontier for 7 layers and 500 iterations

TABLE VII. METRICS FOR 15 LAYERS AND 500 ITERATIONS

	#NS	Hypervolume	Time	Final #NS
M1	65.93	3.27E+15	332.18	139
Stdev	4.46	3.79E+14	13.08	
M2	116.67	2.87E+15	331.33	233
Stdev	11.43	6.70E+14	8.75	
M3	171.77	2.18E+15	328.35	358
Stdev	16.77	6.18E+14	8.24	
M4	191.33	3.71E+15	331.55	379
Stdev	12.99	6.90E+14	8.51	
M5	246.067	3.27E+15	329.20	460
Stdev	14.88	8.72E+14	10.02	

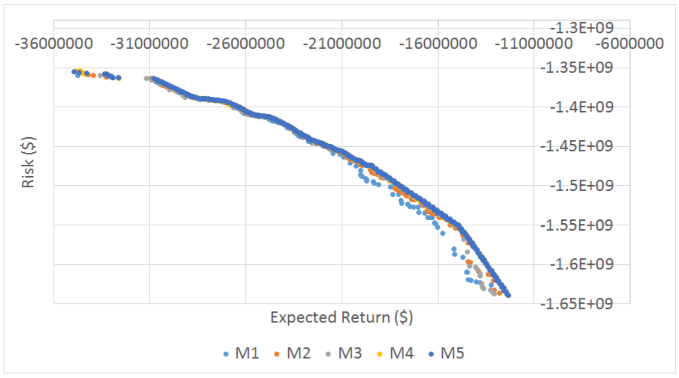

Figure 5. Final Pareto frontier for 15 layers and 500 iterations

TABLE VIII. COVERAGE FOR 15 LAYERS AND 500 ITERATIONS

	M1	M2	M3	M4	M5
M1	-	0.051	0.05	0.005	0.00
M2	0.777	-	0.154	0.0474	0.002
M3	0.77	0.70	-	0.184	0.046
M4	0.964	0.83	0.497	-	0.104
M5	0.99	0.897	0.706	0.547	-

solutions from M1 as shown by Table X. On the other hand, the differences between M4 and M5 are not so substantial because M5 dominates only 13% of solutions from M4 which represents that 87% of the solutions on both sets are either the same or non-dominated solutions.

When we move to 15 layers (Figure 7), a larger number of iterations do not create better solutions for M1. Also, more iterations do not significantly approximate M4 from M5 as we can see in Table XII where M5 dominates 49% of solutions from M4, whereas M4 dominates only 6.3% of solutions from M5. This behavior is a strong indication that M5 is the best operator for solving this problem.

TABLE IX. METRICS FOR 7 LAYERS AND 1000 ITERATIONS

	#NS	Hypervolume	Time	Final #NS
M1	122.83	2.32E+15	412.13	281
	6.80	2.05E+13	9.14	
M2	215.93	2.27E+15	411.85	400
	14.21	2.28E+14	11.16	
M3	199.73	1.88E+15	410.44	353
	20.62	3.75E+14	6.91	
M4	292.57	2.20E+15	413.74	404
	11.19	2.42E+14	8.87	
M5	333.27	2.19E+15	412.37	397
	14.25	2.59E+14	11.29	

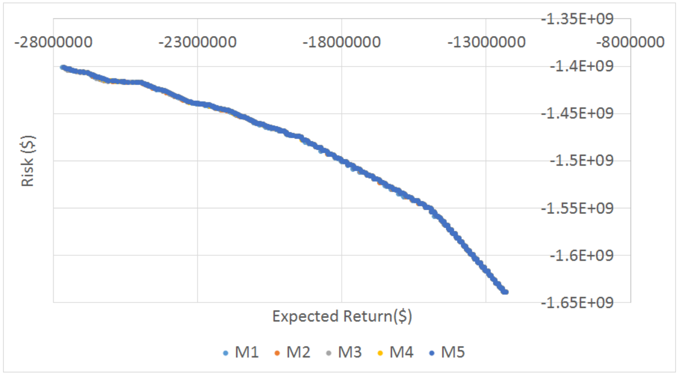

Figure 6. Final Pareto frontier for 7 layers and 1000 iterations

TABLE X. COVERAGE FOR 7 LAYERS AND 1000 ITERATIONS

	M1	M2	M3	M4	M5
M1	-	0.190	0.187	0.131	0.118
M2	0.665	-	0.136	0.029	0.005
M3	0.68	0.240	-	0.066	0.00
M4	0.79	0.305	0.21	-	0.00
M5	0.808	0.350	0.240	0.133	-

TABLE XI. METRICS FOR 15 LAYERS AND 1000 ITERATIONS

	#NS	Hypervolume	Time	Final #NS
M1	75.00	3.36E+15	655.36	147
	6.88	3.00E+14	26.82	
M2	143.43	3.22E+15	656.79	298
	12.18	6.30E+14	23.45	
M3	211.33	2.25E+15	662.09	361
	21.87	6.30E+14	10.68	
M4	242.53	3.67E+15	655.87	439
	13.17	7.48E+14	20.70	
M5	299.37	3.52E+15	679.67	507
	14.33	7.31E+14	29.95	

TABLE XII. COVERAGE FOR 15 LAYERS AND 1000 ITERATIONS

	M1	M2	M3	M4	M5
M1	-	0.067	0.017	0.006	0.006
M2	0.782	-	0.061	0.041	0.012
M3	0.857	0.775	-	0.273	0.065
M4	0.952	0.778	0.243	-	0.063
M5	0.972	0.88	0.46	0.49	-

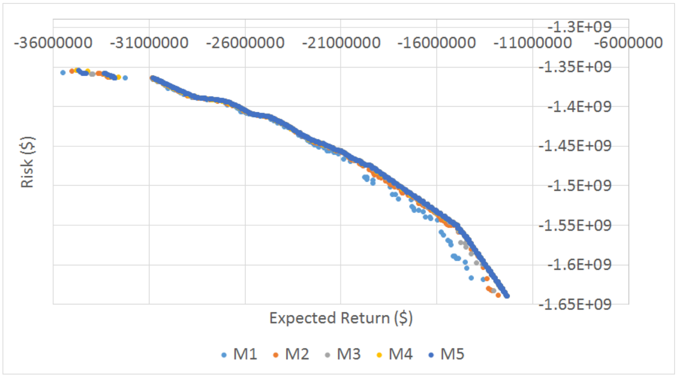

Figure 7. Final Pareto frontier for 15 layers and 1000 iterations

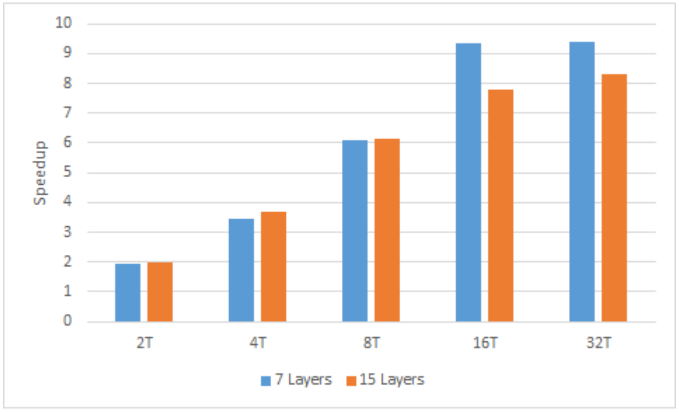

Figure 9. Speedup for 7 and 15 layers and 1000 iterations on Xeon

C. Parallel Version

In order to parallelized the code, we used the Snow package [19] from R which is a package for automatic parallelization. We parallelized the iteration loop using a *foreach* instruction associated with the parameter *%dopar%*. This parameter is responsible for dividing the iterations between threads. The main advantage of this approach is to maintain intact almost the entire code, being necessary to add instructions only for gathering results delivered by each thread. On the other hand, the main disadvantage lays in the fact that as we increase the number of threads the results tend to be worse in terms of quality.

The mutation operator we used is the M5 with 1000 iterations because it presented better results than the other ones. Figures 8 and 9 show the time and speedup reached in the Xeon architecture variating the thread count. Regardless the number of layers, the best efficiency is reached using 2 thread representing an efficiency of 96.7% and 98.2%, respectively. In terms of speedup, it is almost linear up to 4 threads. Then, the best one is reached using 32 threads representing 9.38 and 8.33 for 7 and 15 layers, respectively; however, the use of 32 threads represents an efficiency of 29.3% and 26% for 7 and 15 layers. Moreover, the best speedups are reached by 7 layers saturating in approximately 16 threads.

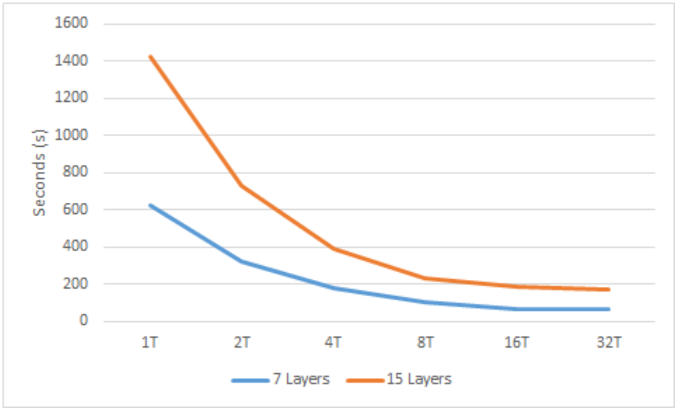

Figure 8. Time for 7 and 15 layers and 1000 iterations on Xeon

Figure 10 presents the Pareto frontier obtained by varying the thread count for 1000 iteration and 7 layers, where we can observe that visually all Pareto frontiers seem to be the same. Table XIII depicts the averages in term of metrics. Even though, the number of solutions decrease as we increase the number of threads, the final number of solutions is not affected. Moreover, the hypervolume is quite stable between threads; therefore, the faster the execution the better. In fact, the small numbers in Table XIV, which represent the coverage, mean that the Pareto frontiers are very similar regardless the number of threads.

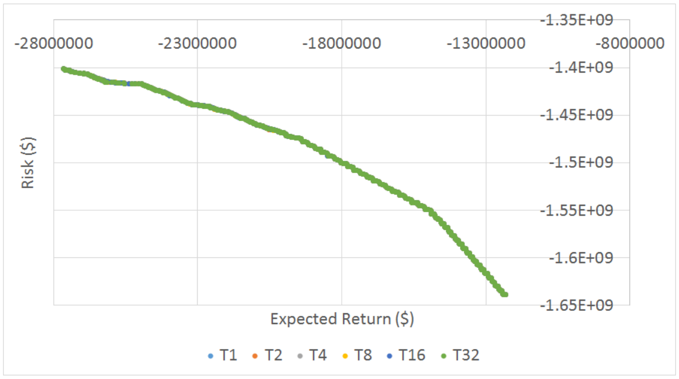

Figure 10. Pareto frontier varying thread count for 1000 iteration and 7 layers

TABLE XIII. METRICS FOR 7 LAYERS AND 1000 ITERATIONS

	#NS	Hypervolume	Time	#NS Final
1T	337.3666667	2.25E+15	626.3866667	403
	12.60673967	2.14E+14	3.61856901	
2T	336.6333333	2.30E+15	323.8888	398
	11.60999371	1.59E+14	1.890259066	
4T	329.6333333	2.34E+15	181.6333667	390
	10.49296425	1.05E+14	0.808078286	
8T	315.8666667	2.35E+15	103.0467333	406
	12.01359383	1.28E+13	0.340365719	
16T	288.9666667	2.35E+15	67.123	403
	9.86628999	3.00E+13	0.711079801	
32T	246.9	2.35E+15	66.7494	390
	13.47628824	1.45E+13	2.711422067	

Figure 11 shows the Pareto frontier obtained by varying the thread count for 1000 iteration and 15 layers, where we can observe that visually the difference between Pareto frontiers obtained by different counting of threads is not meaningful.

TABLE XIV. COVERAGE FOR 7 LAYERS AND 1000 ITERATIONS

	T1	T2	T4	T8	T16	T32
T1	-	0.028	0.03	0.08	0.16	0.20
T2	0.04	-	0.04	0.09	0.16	0.215
T4	0.03	0.03	-	0.07	0.14	0.20
T8	0.030	0.035	0.028	-	0.14	0.17
T16	0.019	0.015	0.015	0.057	-	0.16
T32	0.017	0.022	0.026	0.02	0.086	-

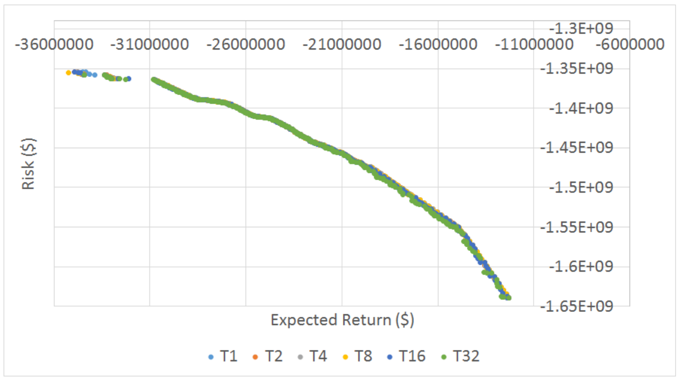

Figure 11. Pareto frontier varying thread count for 1000 iteration and 15 layers

TABLE XV. METRICS FOR 15 LAYERS AND 1000 ITERATIONS

	#NS	Hypervolume	Time	#NS Final
1T	290.73	3.39E+15	1426.90	517
	22.97	8.93E+14	6.33	
2T	296.03	3.82E+15	726.32	515
	15.83	7.22E+14	2.16	
4T	280.37	4.03E+15	387.39	470
	12.70	5.74E+14	1.13	
8T	237.40	4.22E+15	232.96	450
	13.63	3.72E+14	1.12	
16T	201.00	4.12E+15	182.87	378
	13.43	3.30E+14	9.25	
32T	164.00	3.99E+15	171.36	336
	12.71	4.05E+14	16.34	

TABLE XVI. COVERAGE FOR 15 LAYERS AND 1000 ITERATIONS

	T1	T2	T4	T8	T16	T32
T1	-	0.50	0.57	0.68	0.79	0.87
T2	0.40	-	0.32	0.54	0.65	0.77
T4	0.30	0.14	-	0.46	0.61	0.75
T8	0.20	0.09	0.15	-	0.52	0.70
T16	0.13	0.05	0.10	0.16	NA	0.55
T32	0.059	0.017	0.04	0.08	0.20	NA

Table XV presents data showing the relationship between the number of solutions and the number of threads used. As we increase the number of threads the number of solutions decreases. Overall, 8 threads seems to be a sweet spot due to the size of the associated hyper volume combined with the reduction in running time. Table XVI reinforces this view in that the 8 thread solution dominates 52% and 70% of solutions using 16 and 32 threads, respectively.

V. Conclusion

This paper presented a modified version of a DE algorithm for multi-objective algorithms with application in reinsurance analytics. Five mutations operators were tested. Results indicated that the best one is called M5, where the first element of the mutation operator is chosen from the archive. Parallel speedup experiments were performed on a Xeon based multi-core machines achieving a speedup of 9.38 using 32 threads.

Acknowledgment

The authors would like to thank NSERC, CNPq and IFMA for funding this research.

References

[1] J. Branke, K. Deb, K. Miettinen, and R. Sowiski, ”Introduction to Multiobjective Optimization: Interactive and Evolutionary Approaches“, Lecture Notes in Computer Science (LNCS), vol. 5252, 2008.

[2] O. A. C. Cortes, A. Rau-Chaplin, D. Wilson, I. Cook, and J. Gaiser-Porter, “Efcient Optimization of Reinsurance Contracts using Discretized PBIL”, DATA ANALYTICS, Porto-Portugal, 2013, pp. 18–24.

[3] O. A. C. Cortes, A. Rau-Chaplin, D. Wilson, and J. Gaiser-Porter, “On PBIL, DE and PSO for Optimization ofReinsurance Contracts”, EvoStar, EvoFin, LNCS, Barcelona, 2014, pp. 227–238.

[4] J. Cai, K. N. Tan, C. Weng, and Y. Zhang, “Optimal reinsurance under VaR and CTE risk measures”. Insurance: Mathematics and Economics, 43, 2007, pp. 185–196.

[5] R. Storn and K. Price, “Minimizing the real functions of the ICEC96 contest by differential evolution”, Proc. of IEEE International Conference on Evolutionary Computation, Nagoya, Japan, 1996, pp. 842–844.

[6] M. J. Reddy and D. N. Kumar, “Multiobjective Differential Evolution with Application to Reservoir System Optimization”, Journal of Computing on Civil Engineering, no. 21, 2007, pp. 136–146.

[7] S. P. Sotiroudis, S. K. Goudos, K. A. Gotsis, K. Siakavara, and J. N. Sahalos, “Application of a Composite Differential Evolution Algorithm in Optimal Neural Network Design for Propagation Path-Loss Prediction in Mobile Communication Systems”, Antennas and Wireless Propagation Letters, IEEE, vol. 12, 2013, pp. 364–367.

[8] H. Zhou and J. Zhang, “Application of Differential Evolution Optimization Based Gaussian Mixture Models to Speaker Recognition”, The 26th Chinese Control and Decision Conference (2014 CCDC), 2014, pp. 4297–4302.

[9] A. F. Shapiro and R. P. Gorman, “Implementing adaptive nonlinear models”, Insurance: Mathematics and Economics, vol. 26, Issues 23, 2000, pp. 289–307.

[10] P. Posík, W. Huyer, and A. Pál, “A comparison of global search algorithms for continuous black box optimization”, Evolutionary Computation, vol. 20, 2012, pp. 509–541.

[11] S. Salcedo-Sanz, L. C. Calvo, M. M. C. Bielsa, A. Castañer, and M. Marmol, “An Analysis of Black-Box Optimization Problems in Reinsurance: Evolutionary-Based Approache”. Available at SSRN: http://ssrn.com/abstract=2260320 or http://dx.doi.org/10.2139/ssrn.2260320, 2013, [Retrieved: May-2015]

[12] O. A. C. Cortes, P. F. do Prado, and A. Rau-Chaplin, “On VEPSO and VEDE for Solving a Treaty Optimization Problem”, Data Analytics, IEEE International Conference on System, Man, and Cybernetics, Sandiego-USA, 2014, pp. 2427–2432.

[13] T. Robič and B. Filipič,“DEMO: Differential Evolution for Multi-objective Optimization”, 3rd International Conference on Evolutionary MultiCriterion Optimization, LNCS, 2005, pp. 520-533.

[14] M. Ali, P. Siarry, and M. Pant, “An efficient Differential Evolution based algorithm for solving multi-objective optimization problems”, European Journal of Operational Research, Volume 217, Issue 2, 2012, pp. 404–416.

[15] Deb. K., “Multi-objective Optimization using Evolutionary Algorithms”, John Wiley and Sons LTDA, 2001.

[16] A. K. Qin, V. L. Huang, and P. N. Suganthan, “Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization”, IEEE Transactions on Evolutionary Computation, vol. 13, no.2, 2009, pp. 398–417.

[17] Q. Zhang and H. Li, “MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition”, IEEE Transactions on Evolutionary Computation, vol. 11, no. 6, 2007, pp.712–731.

[18] E. Alba, “Parallel Evolutionary Algorithms Can Achieve Super-Linear Performance”, Information Processing Letters, vol. 82, 2002, pp. 7-13.

[19] Snow Package, http://cran.r-project.org/web/packages/snow/index.html, [Retrieved: May-2015]

[20] H. Wang, O. A. C. Cortes, A. Rau-Chaplin, “Dynamic Optimization of Multi-layered Reinsurance Treaties”, Symposium on Applied Computing, ACM, Salamanca-Spain, 2015, pp. 125–132.