On VEPSO and VEDE for Solving a Treaty Optimization Problem

Omar Andres Carmona Cortes*, Andrew Rau-Chaplin†, Pedro Felipe do Prado‡
*Informatics Academic Department
Federal Institute of Science, Education and Technology of Maranhão
omar@ifma.edu.br
†Risk Analytics Lab
Dalhousie University
arc@cs.dal.ca
‡Institute of Mathematics and Computer Science
University of São Paulo
pfprado@icmc.usp.br

***Abstract*—The purpose of this paper is to evaluate the performance of Vector Evaluated Differential Evolution (VEDE) and Vector Evaluated Particle Swarm Optimization (VEPSO) in solving a real world financial optimization problem. The algorithms have been applied to the Reinsurance Contract Problem, which is a challenging problem in computational finance, and their performance has been evaluated in terms of metrics including the average number of solutions, the average hypervolume and the coverage. Results have shown that both algorithms can reach good solutions, however VEPSO tends to perform better.**

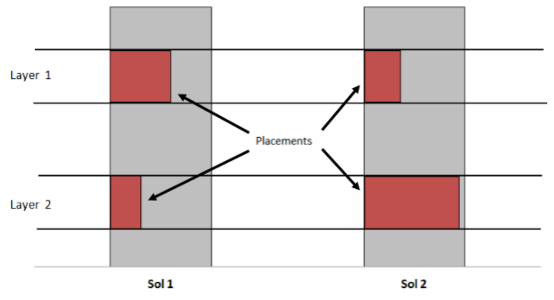
I. INTRODUCTION

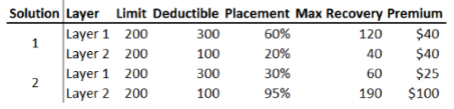
The vector evaluated approach for multiobjective optimization was pioneered by Shaffer [1] in 1985 in the context of Vector Evaluated Genetic Algorithms (VEGA). At that time, GAs were a popular form of optimization, while algorithms such as Particle Swarm Optimization (PSO) and Differential Evolution had yet to be introduced. Recently, Matthysen et. al. [2] has shown that VEGA tends to get trapped into local optima. Therefore, it is interesting to explore the vector evaluated approach in the context of more recent evolutionary strategies.

VEPSO and VEDE were proposed by Parsopoulus in [3] and [4], respectively. Since then, their applications have included optimization of radiometry array antenna [5], production scheduling [6], steady-state performance of a power system [7], and energy preservation in communication systems [8].

In this work, we have evaluated VEPSO as well as VEDE approaches for solving a specific treaty optimization problem called Optimization of Reinsurance Contracts (ORC), which, from the perspective of the insurance company, consist of given a main structure of layers, their limits, deductibles, recoveries and premiums, we have to find out the best combination of shares or placements which hedge the maximum risk to a second insurance company (a reinsurance company) and at the same time maximize the amount of money received in case of massive claims (expected return). Figure 1 represents a structure and two particular solutions.

Cortes in [9] and [10] presented the first papers to tackle this type of problem using evolutionary algorithms, including Population-Based Incremental Learning (PBIL), PSO and DE. While these methods performed significantly better than the previous exhaustive enumeration approach they suffered from one significant drawback. They were based on a mono-objective function, where the maximum risk was obtained only for a given expected return. So in order to generate a Pareto frontier the algorithms had to be executed many times resulting in a method that was sometime too slow to be used on industrial scale problems. A key advantage of the true multiobjective approaches described in this paper is that they generate the whole Pareto frontier at once.





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

Fig. 1. Structure and two solutions with different placements.

The remainder of this paper is organized as follows: Section II introduces both the fundamentals of multiobjective problems and the treay optimization problem being solved in this paper; Section III outlines the algorithms we have been used; Section IV presents metrics, parameters and the performance evaluation; finally, Section V shows the conclusions of this works and intended works.

II. MULTIOBJECTIVE PROBLEMS

A multiobjective optimization problem (MOP) has to address two or more conflicting objective function [11] 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.

Thus, assuming that a solution to a MOP is a vector in a search space X with m elements. A function $f : X \rightarrow Y$ evaluates the quality of a solutions mapping it into an objective space. Therefore, a multi-objective problem is defined as presented in Equation 1, where f is a vector of objective functions, m is the dimension of the problem and n 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, 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

The reinsurance contract optimization problem is a particular kind of treaty optimization problem, which 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 [9]. Taking this into consideration, the task is to identify optimal combinations of shares (also called placements) in order to build a Pareto frontier that quantifies the best available trade-offs between expected return and risk. In other words, insurance companies aim hedge their risk against potentially large claims, or losses [12]. Having these trade-offs the insurance companies are able to offer them to the reinsurance market.

All in all, the purpose is both to maximize the amount of return ($) received from the reinsurance company and maximize the risk hedge to it, at the same time. Doing so, the insurance company minimize the loss faced per year. In this context, Equation 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[1]. For further details about the problems refer to [9] and [12].

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

[1]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.

III. Vector Evaluated Algorithms

As mentioned before, VEGA was the first vector evaluated-based algorithm. The main idea is to evolve different populations, one per evaluation function, then exchange information between them.

Extending this idea to other algorithms is a natural approach. The popularity of the vector evaluated approach comes from two advantages: it is easy to implement and it can be parallelized in a straight-full manner [4].

A. VEPSO

VEPSO "evolves" two independent swarms and exchange information between them. Basically, the best particle of one swarm directs the evolution of the other one. The pseudo code is presented in the Algorithm 1 considering a minimization problem.

```
1  forall the Swarms $\in k$ do
2      $S^k \leftarrow$ generate_swarm($f_k$,n,d)
3      $P^k \leftarrow S^k$
4      $V^k \leftarrow$ init_velocity(n,d,$V_{min}$,$V_{max}$)
5      $fit^k \leftarrow$ evaluate_$f_k(S^k)$
6      $fitP^k \leftarrow fit^k$
7      $fitg^k \leftarrow$ best($fit^k$)
8      $g^k \leftarrow$ locate_best_particle($S^k$,$fitg^k$)
9      for iter = 1 to #iterations do
10         for i = 1 to #swarm_size do
11             $V_i^k = w*V_i^k+c_1*r_1*(P_i^k-S_i^k)+c_2*r2*(g^l-S_i^k)$
12             $S_i^k = S_i^k + V_i^k$
13             $fit_i^k = evaluate(S_i^k)$
14             if ($fit_i^k < fitP_i^k$) then
15                 swap($fitP_i^k$,$fit_i^k$)
16                 swap($P_i^k$,$S_i^k$)
17             end
18             if ($fit_i^k < fitg^k$) then
19                 swap($fitg^k$,$fit_i^k$)
20                 swap($g^k$,$S_i^k$)
21             end
22         end
23         archive = join($S^k$,$S^{k+1}$)
24         $fit_2^l \leftarrow$ evaluate_$f_2(S^l)$
25         $fit_1^l \leftarrow$ evaluate_$f_1(S^l)$
26         archive_fit $\leftarrow$ join($fit^1$,$fit_2^1$,$fit_1^2$,$fit^2$)
27         pareto $\leftarrow$ evaluate_domination(archive_fit)
28         pareto_pop $\leftarrow$ n_dominated(indexes,archive)
29     end
30 end
```
Algorithm 1: VEPSO

Considering two swarms S^1 and S^2, the algorithm starts initializing each swarm with n particles of dimension d. The matrix P^k, $k = 1, 2.$, stores the best position where each particle has passed by. V^k, $k = 1, 2.$, represents the velocity which is used to change the position of a particle in the search space. The vector fit^k stores the fitness of the current particle position. The structure $fitP^k$ is a vector containing the fitness for each line from P^k. The global optima is saved in fit_g^k and its respective position is stored in g^k.

On each swarm and iteration, the velocity of each particle is updated using the equation $V_i^k = w * V_i^k + c_1 * r_1 * (P_i^k - S_i^k) + c_2 * r2 * (g^l - S_i^k)$, then the position is updated using $S_i^k = S_i^k + V_i^k$, where i is an index which represents a line on both V and S matrix, *i.e.*, these equations are vector-based operations, and k indicates in which swarm the operations are being done. Afterwards, the fitness of each new particle position is evaluated and compared against its historic position in P, if the new fitness is better than the known one the algorithm swaps them. Thereafter the same operation has to be done in order to verify whether a new global best was achieved or not. At the end of each iterations, the fitness of each particle is evaluated in the opposite function, for example, the swarm S^1 is evaluated in terms of the function f_2, and vice-versa. This operation allows the algorithm to construct and maintain the archive whose the aim is to store the Pareto frontier and its respective positions in the search space.

B. VEDE

The idea behind Vector Evaluated Differential Evolution is similar to the previous algorithms, i.e., evolve to independent populations and share information between them. The Algorithm 2 outlines how the VEDE works.

Firstly, the algorithm initializes the populations and their fitness. Then on each iteration and for each individual in population (also called target individuals) the algorithm executes the mutation process, which is to obtain the vector of differences between three individuals selected randomly according to the equation $v \leftarrow P_{idx[3]}^k + F * (Pop_{idx[1]}^k - Pop_{idx[2]}^k)$. This startegy is named $DE/Rand/1$. Afterwards the crossover process in started, which is very similar to the one called discrete crossover performed in genetic algorithms, where each individual is created by selecting a gene from the vector of differences (v) or form the current population. This choice is made based on the crossover rate (CR), then if a random number is less than CR the gene used in the new individual comes from v_j, otherwise it comes from an individual from the current population pop_{ij}.

After coming up with the new individual its fitness is evaluated. If the new fitness is better than the target one in the current population the new individual replaces it, otherwise the target is maintained in the population. Giving the new population, it is necessary to identify the best solution of each population (named sol_1 and sol_2, representing $indiv_1$ and $indiv_2$ from population P^1 and P^2, respectively) and compare them against each other in terms of dominance. If sol_1 dominates sol_2 then $indiv_1$ replaces $indiv_2$ in P^2, and vice-versa otherwise. Then the archive is updated in the same way as in VEPSO.

A secondary VEDE approach was also implemented in order to make VEDE more similar to VEPSO. In this case, lines ranging from 21 to 25 were removed and the best individual from a particular population is used to compute the vector of differences, i.e., $v \leftarrow Pop_{idx[1]}^k + F * (best^1 - Pop_{idx[1]}^k) + F * (best^2 - Pop_{idx[2]}^k)$, where $best^1$ represents the best solution in terms of the current population and $best^2$ depicts the best solution coming from the other population. This strategy is called $DE/best/2$ and eliminates the necessity of comparing the dominance of best solutions belonging to P^1 and P^2.

```
1  forall the $Pop \in k$ do
2      $P^k \leftarrow$ generate_pop($f_k$,n,d)
3      $fit^k \leftarrow$ evaluate_$f_k(P^k)$
4      for iter = 1 to #iterations do
5          for i = 1 to #pop_size do
6              $idx \leftarrow$ select_indiv(3)
7              $v \leftarrow P_{idx[3]}^k + F * (Pop_{idx[1]}^k - Pop_{idx[2]}^k)$
               for j = 1 to dimension do
8                  nj = rand()
9                  if (nj < CR) then
10                     pop'$\leftarrow v_j$
11                 else
12                     pop'$\leftarrow pop_i j$
13                 end
14             end
15             $fit'_i \leftarrow$ evaluate_$f_k(P_i^k)$
16             if $fit'_i < fit_i$ then
17                 $pop_i \leftarrow pop'_i$
18                 $fit_i \leftarrow fit'_i$
19             end
20         end
21         $sol_k \leftarrow get\_best(P^k)$
22         if ($sol_1$ dominates $sol_2$)) then
23             replace($indiv_1 \in P^2$ )
24         else if ($sol_2$ dominates $sol_1$)) then
25             replace($indiv_2 \in P^1$)
26         archive $\leftarrow$ join($P^1$,$P^2$)
27         $fit_2^1 \leftarrow$ evaluate_$f_2(S^1)$
28         $fit_1^2 \leftarrow$ evaluate_$f_1(S^2)$
29         archive_fit $\leftarrow$ join($fit^1$,$fit_2^1$,$fit_1^2$,$fit^2$)
30         pareto $\leftarrow$ evaluate_domination(archive_fit)
31         pareto_pop $\leftarrow$ n_dominated(pareto,archive)
32     end
33 end
```

Algorithm 2: VEDE

IV. Experiments

A. Metrics

In this section, we discuss the experimental evaluation of the vector evaluated-based algorithms. Firstly, the number of non-dominated points (number of solutions) found in the Pareto frontier was determined. Secondly, the hypervolume, which is the volume of the dominated portion of the objective space as presented in Equation 3, was measured, where for each solution $i \in Q$ a hypercube v_i is constructed. Having each v_i, we calculated the final hypervolume by the union of all v_i.

$$hv = volume(\bigcup_{i=1}^{|Q|} v_i) \tag{3}$$

Thirdly, the dominance relationship between Pareto frontiers obtained with different algorithms, *i.e.*, the coverage, was calculated as depicted in Equation 4. Roughly speaking, $C(A, B)$ is the percentage of the solutions in B that are dominated by at least 1 solution in A [13], 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 $C(A, B) \neq C(B, A)$.

$$C(A, B) = \frac{|\{b \in B | \exists a \in A : a \preceq b\}|}{|B|} \tag{4}$$

For further details about the use of these metrics see [11]. Finally, the resulting frontiers can be reviewed by experts for reasonability.

B. Statistics and Parameters

All experiments have been executed using 100, 500 and 1000 iterations, and 31 trials for each configuration. We chose this number of runs because of the central limit theory which roughly states that 30 runs is the minimum number of repetition in order to consider a distribution as a normal one, allowing us to use parametric tests as Analysis of Variance (ANOVA) and Tukey test. So, considering a null hypotheses (h_0) as "all means presented by the algorithms are statically the same" and a level of significance $\alpha = 0.05$, we use the ANOVA test in order to identify whether h_0 is true or not. In such test if F is outside of the boundaries $[-F_{crit}, F_{crit}]$ then the difference exist, *i.e.*, we reject h_0. Even though the ANOVA test might identify that there are differences between means, the test is not able to detect where the differences are. In this case, we use a Tukey Test to identify it, comparing the algorithms in pairs according to Equation 5, where M_1 is the average of the first sample, M_2 is the average of the second sample, MS is the error between groups from the respective ANOVA test and n is the number of trials ($n = 31$ in our case). If HSD is outside from the range $[-q, q]$, obtained from the Tukey's significance/probability table, then there is a difference in this particular pair of samples.

$$HSD = \frac{M_1 - M_2}{\sqrt{\frac{MS}{n}}} \tag{5}$$

All the parameters were chosen empirically and all tests have been 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 16 GB of RAM.

The parameters used in all experiments for PSO algorithms were: $c_1 = c_2 = 0.5 + log(2)$; numbers of particles = 50; w_0 = 0.9; w_f = 0.1, where w has a linear updating based on the Equation 6 as proposed in [14], where w_0 is the initial weight, w_f is the final one, N_G is the number of iterations and i depicts the current generations. The initialization was done as recommended in [15]; and, regarding the parameters for DE, they are $f = 0.7$, $CR = 0.9$, population size = 50, and the strategies $DE/Rand/1$ (called VEDE-R) and $DE/Best/2$ (called VEDE-B).

$$w = (w_0 - w_f) \times \frac{N_G}{i} \tag{6}$$

The final Pareto frontier after all executions is also shown at the end of the experiment section. Moreover, we are not narrowing the size of the archive, therefore we can compare the average number of solutions. Data came from a real industry, was anonymized, comprising of 7 layers and we used a discretization level of 5%, *i.e.*, all combination of shares will range from 0 to 1 with a variation of 0.05. The discretization level is a market requirement and normally can be set as 1%, 5%, 10% and 25%. The last two discretization options can be used by exhausted search algorithms, whereas 1% and 5% can be solved only by meta-heuristics. Figure 2 shows an estimation of the time required for solving the problem by an exhaustive search algorithms, where the entire space is discretized then the solutions are evaluated. Thus, we can see that using the same parameters set in our work, the exhaustive method would take much more than a week to solve the problem. Actually, considering the same architecture and parameters the real estimation would take 1 year to complete.

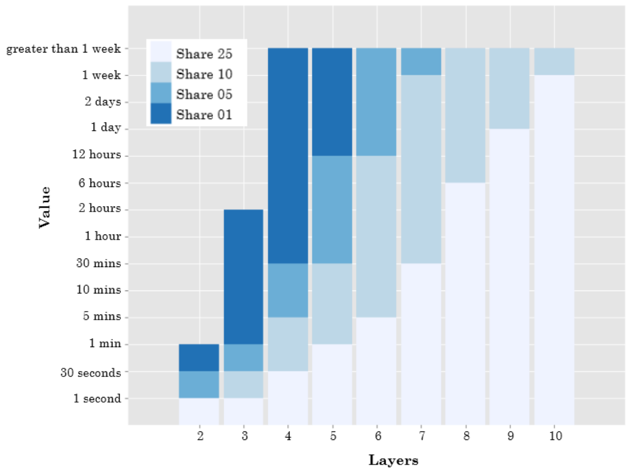


Fig. 2. Estimation of time required by the exhaustive search algorithm.

C. Performance Evaluation

Figure 3 shows the average time in seconds for 100, 500 and 1000 iterations of each algorithm. According to the figure, the difference starts being noticeable from 1000 iterations forward; however, an ANOVA test presented in Table I indicates that the difference in terms of the execution time is significant using 500 iterations, as well. The Tukey test presented in Table II proves that there are differences in all algorithms we have tested because all HSD are outside from the range $[-3.47, 3.47]$, meaning that VEPSO is the fastest approach.

Figure 4 depicts the average number of solutions obtained by each algorithm considering 100, 500 and 1000 iterations. The vertical axis is in a Log10 scale, otherwise the data from 100 and 500 iterations would not be noticed. Clearly, the differences between VEDE and VEPSO seem significant. In fact, Table III identifies that there are differences between the algorithms, and Table IV proves that, giving to VEPSO the best average amount of non-dominated points per trial. Further, we can observe in the tukey test table (IV) that

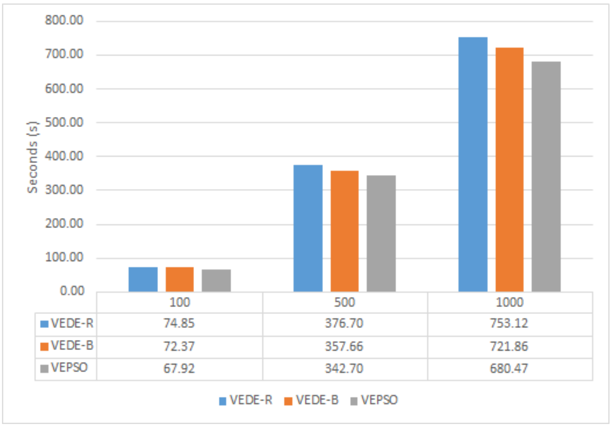


Fig. 3. Average execution time for 100, 500 and 1000 iterations.

TABLE I. ANOVA CONSIDERING THE EXECUTION TIME.

100 iterations						
Source	SS	df	MS	F	P-value	F-crit
Between Groups	763.35	2	381.67	**121.21**	2.9e-26	3.1
Within Groups	283.40	90	3.15			
500 iterations						
Source	SS	df	MS	F	P-value	F-crit
Between Groups	18003.3	2	9001	**125.66**	8.9e-27	3.1
Within Groups	6447.3	90	71.64			
1000 iterations						
Source	SS	df	MS	F	P-value	F-crit
Between Groups	82342.5	2	41171.25	**289.66**	6.13e-40	3.1
Within Groups	12792.32	90	142.14			

TABLE II. TUKEY TEST CONSIDERING THE EXECUTION TIME.

	q=3.47		
	100 iterations	500 iterations	1000 iterations
VEDE-R vs VEDE-B	7.79	12.52	14.6
VEDE-R vs VEPSO	21.73	22.37	33.92
VEDE-B vs VEPSO	13.94	9.84	19.33

strategies $DE/Rand/1$ and $DE/Best/2$ presented similar results regardless the number of iterations.

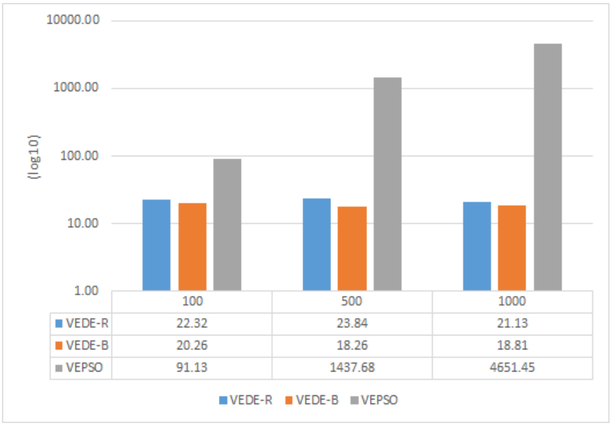


Fig. 4. The average number of solutions for 100, 500 and 1000 iterations.

Regarding the hypervolume, Figure 5 shows the average hypervolume achieved by each algorithm using 100, 500 and 1000 iterations, where we can see that VEPSO got the better average hypervolume. Indeed, Table V confirms that the difference between hypervolumes is significant. Furthermore, Table VI indicates that there are no differences between

TABLE III. ANOVA CONSIDERING THE NUMBER OF SOLUTIONS.

100 iterations						
Source	SS	df	MS	F	P-value	F-crit
Between Groups	100866.6	2	50433.3	**538.4**	8.4e-51	3.1
Within Groups	8430.2	90	93.67			
500 iterations						
Source	SS	df	MS	F	P-value	F-crit
Between Groups	41475131	2	20737565	**937.2**	5.6e-61	3.1
Within Groups	1991461	90	22127.34			
1000 iterations						
Source	SS	df	MS	F	P-value	F-crit
Between Groups	4.43e08	2	2.22e08	**2593.11**	2.73e-80	3.1
Within Groups	7693118	90	85479.1			

TABLE IV. TUKEY TEST CONSIDERING THE NUMBER OF SOLUTIONS.

	q=3.47		
	100 iterations	500 iterations	1000 iterations
VEDE-R vs VEDE-B	1.19	0.20	0.044
VEDE-R vs VEPSO	-29.58	-52.92	-88.17
VEDE-B vs VEPSO	-40.77	-53.13	-88.22

VEDE strategies ($DE/Rand/1$ and $DE/Best/2$) when 1000 iterations are considered; however, this difference is significant when VEDE is compared against VEPSO, where this last one presented the best average hypervolume.

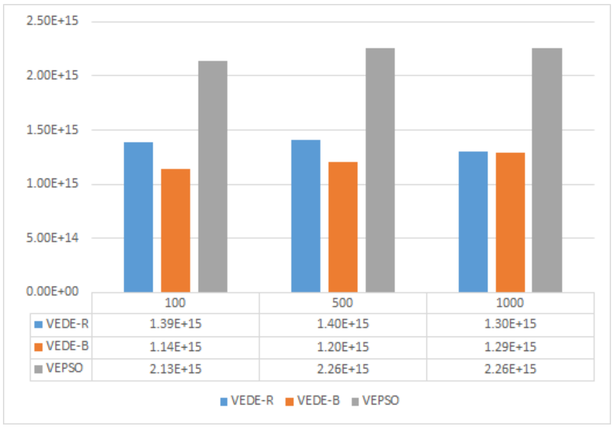


Fig. 5. The average hypervolume for 100, 500 and 1000 iterations.

TABLE V. ANOVA CONSIDERING THE HYPERVOLUME.

100 iterations						
Source	SS	df	MS	F	P-value	F-crit
Between Groups	1.65e31	2	8.26e30	**245.68**	3.47e-37	3.1
Within Groups	3.03e30	90	3.36e28			
500 iterations						
Source	SS	df	MS	F	P-value	F-crit
Between Groups	1.94e31	2	9.7e30	**465.66**	3.38e-48	3.1
Within Groups	1.87e30	90	2.08e28			
1000 iterations						
Source	SS	df	MS	F	P-value	F-crit
Between Groups	1.92e31	2	9.61e30	**367.72**	4.9e-44	3.1
Within Groups	2.35e30	90	2.61e28			

TABLE VI. TUKEY TEST CONSIDERING THE HYPERVOLUME.

	q=3.47		
	100 iterations	500 iterations	1000 iterations
VEDE-R vs VEDE-B	7.6	7.7	0.38
VEDE-R vs VEPSO	-22.53	-32.95	-33.02
VEDE-B vs VEPSO	-30.14	-40.61	-33.40

Figure 6 presents the final Pareto frontier for the 7 layers problem after 31 trials. Axis present negative numbers because it is a matter of hedging risk. Better results should go toward zero located in the upper right corner. Using 100 iterations it is clear that VPSO achieves better results than both VEDE approaches. Considering 500 and 1000 iterations the VEDE approaches improve their result; nonetheless, VEPSO seems be better. In this context, the coverage metrics becomes essential in order to make an accurate evaluation. Thus, Table VII shows the coverage metrics which, therefore, proves that VEPSO is the better approach. Taking this into account we can observe that VEPSO dominates 85% and 81% of VEDE-R and VEDE-B approaches with 100 iterations, respectively. In the remaining results, VEPSO still finding the best Pareto frontier. Extending the analysis we can notice that the approach VEDE-B tends to be better than VEDE-R.

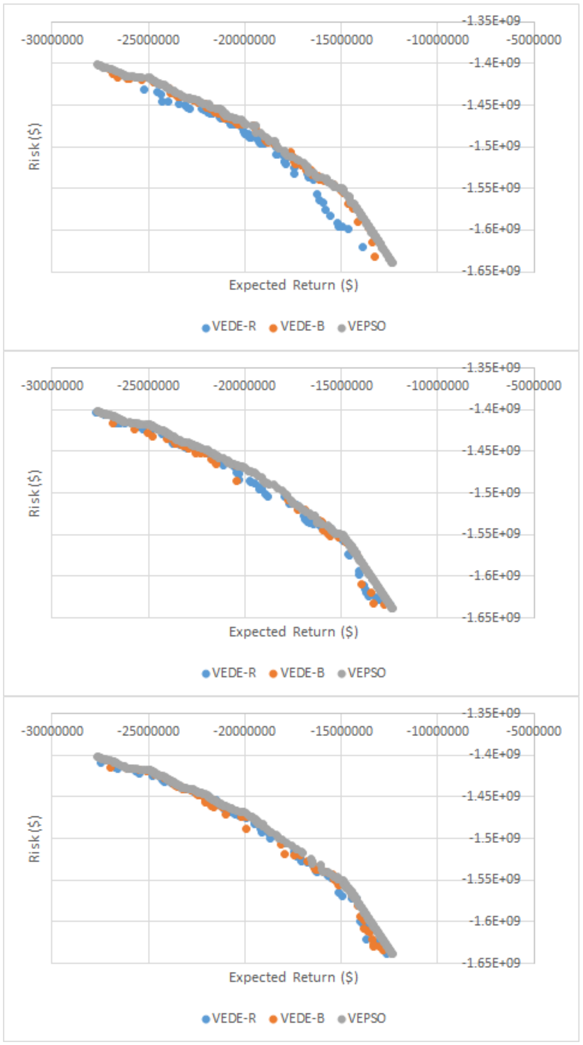


Fig. 6. Final Pareto frontier for 7 layers using 100, 500 and 1000 iterations, respectively.

V. Conclusions

This paper presented a new study about the use of three vector evaluated approaches (VEDE-R, VEDE-B and VEPSO) in the ORC problem. Results clearly show that VEPSO achieves the best results in this particular application, taking approximately 5 hours to find out 90 points in the Pareto frontier. Future work includes: tests in other applications, comparison against VEGA and other multiobjective algorithms, hybridizing VEDE with other multiobjective evolutionary approaches and/or fuzzy in order to improve its results.

TABLE VII. Coverage for VEDE-R, VEDE-B and VEPSO.

100 iterations			
	VEDE-R	VEDE-B	VEPSO
VEDE-R	-	0.28	0.03
VEDE-B	0.5	-	0.06
VEPSO	0.85	0.81	-
500 iterations			
	VEDE-R	VEDE-B	VEPSO
VEDE-R	-	0.31	0
VEDE-B	0.51	-	0.008
VEPSO	0.96	0.87	-
1000 iterations			
	VEDE-R	VEDE-B	VEPSO
VEDE-R	-	0.28	0.02
VEDE-B	0.5	-	0.06
VEPSO	0.85	0.81	

References

[1] J. D. Schaffer, “Multiple objective optimization with vector evaluated genetic algorithms” . In Proc. of First International Conference on Genetic Algorithms, pp. 93-100,1985.

[2] W. Matthysen and A. P. Engelbrecht and K. M. Malan, “Analysis of stagnation behavior of vector evaluated particle swarm optimization”, IEEE Symposium on Swarm Intelligence (SIS), pp.155,163, 2013.

[3] K. E. Parsopoulos and M. N. Vrahatis, “Particle Swarm Optimization Method in Multiobjective Problems”, In Proceedings of the 2002 ACM Symposium on Applied Computing, pp. 603–607, ACM Press, 2002

[4] Parsopoulos, K.E.; Tasoulis, D.K.; Pavlidis, N.G.; Plagianakos, V.P.; Vrahatis, M.N., “Vector evaluated differential evolution for multiobjective optimization”, Congress on Evolutionary Computation, pp.204-211, 2004.

[5] D. Glies and Y. Rahmat-Samii, “Vector evaluated particle swarm optimization (VEPSO): Optimization of a radiometer array antenna”, in Proc. of the IEEE International Symposium on Antennas and Propagation,vol. 3, Jun. 2004, pp. 2297-2300.

[6] Grobler, J. and Engelbrecht, A.P. and Yadavalli, V. S S, “Multi-objective DE and PSO strategies for production scheduling”, IEEE Congress onEvolutionary Computation, pp.1154-1161, 2008.

[7] J. G. Vlachogiannis and K. Y. Lee, “Multi-objective based on parallel vector evaluated particle swarm optimization for optimal steady-state performance of power systems”, Expert Systems with Applications, vol. 36, no. 8, pp. 802-808, 2009.

[8] S. Hou and X. Zhang and H. Zheng and L. Zhao and W. Fang, “An effective interference management framework to achieve energy-efficient communications for heterogeneous network through cognitive sensing”,International ICST Conference on Communications and Networking in China (CHINACOM), pp.536,541, 2012.

[9] O. A. C. Cortes and A. Rau-Chaplin and D. Wilson and I. Cook and J. Gaiser-Porter, “Efcient Optimization of Reinsurance Contracts using Discretized PBIL”, Data Analytics, Porto, 2013.

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

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

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

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

[14] A. Nikabadi and M. Ebadzadeh , Particle swarm optimization algorithms with adaptive Inertia Weight : A survey of the state of the art and a Novel method, IEEE journal of evolutionary computation , 2008.

[15] M. Clerc, “Standard Particle Swarm Optimisation”, hal-00764996, version 1, available in: http://hal.archives-ouvertes.fr/hal-00764996, 2012.