Relay Optimization Method

Frank Xuyan Wang

Abstract—**Insurance-linked securities portfolio with the VaR constraint optimization problem have a kind of weak dominance or ordering property, which enables us to reduce the variables' dimensions gradually through exercising a genetic algorithm with randomly selected initial populations. This property also enables us to add boundary attraction potential to GA-MPC's repair operator, among other modifications such as using Sobol sequence in the initial population selection for multiple runs, using WELL algorithm for RNG, adding orthogonal crossover to the mutation operator, using opposition-based starting population, using clockwise-shaped searching vectors instead of the original counterclockwise-shaped searching vectors, adding the catfish and pivot algorithms for combating the stagnation, and using slowly-changing betas in the searching vector coefficients, resulting in a hybridized GA-MPC-OX algorithm. Numerical experiment shows that the evolved GA-MPC-OX algorithm used in our relay optimization process gives the best objective values for our specific real world optimization problem, among more than twenty other algorithms.**

I. Introduction

AN insurance-linked portfolio consists of weighted reinsurance contracts, in which each contract has a limit in proportion to the weights; the weighted sum of these limits is the portfolio limit, similarly with contract and portfolio premium definition [1]. We also have a universe of loss scenarios, each scenario usually has a constant probability of occurrence, for each contract; the weighted sum of these losses are the portfolio losses. The risk limit is inflicted upon the quantiles of the portfolio losses, also called VaRs. The VaRs are piecewise linear zigzag functions with respect to the weights and nonconvex. The optimization objective is to maximize the portfolio expected profit, or the premium subtracted by the loss, subject to risk limit constraints and other constraints, such as portfolio limit or individual contract limit bounds.

A related but simpler problem is replacing the VaR constraints with TVaR constraints, the mean of all the VaRs above a given quantile. The TVaR constraint is a convex function, which will guarantee that most algorithms converge to the unique global solution. In the VaR case, if we combine the risk limit violation into the expected profit as a penalty term, we can see the objective value function has many very narrow basins surrounded by high mountains, acting as suboptimal traps that prevent many algorithms from surmounting to the global optimum.

This nonconvex and the high dimensionality (corresponding to the number of contracts) present a challenging problem, but also a good tool for testing the publicly available algorithms. The algorithms tested well by diverse benchmark problems are starting candidate algorithms, but they are not guaranteed to work well on our specific problem domain. On one hand, we need to find our problem's domain specific properties, and the algorithms that can make good use of these properties. On the other hand, we need to hybridize and evolve the algorithms so that by the end, we can arrive at the ones which work best.

In using the stochastic algorithm, the result usually cannot repeat and contradictory results are commonly observed. The usual deduction logic does not work. We need a kind of "noise reasoning", a para-theoretical analysis (for the numerical experiment design) by guesswork, symmetry, preference, and metaphors, from social or history wisdom, or sports competitions and other competition strategies.

For example, if we want to form a first place champion candidate team, we will not select those individuals who are stable, but rather those who have the potential or showed traces of burst extraordinary performance capability. But if our goal is the total number of medals, which it is not, we would want more stable candidates that perform on the higher quantiles region of the record spectrum. Also, if a candidate failed several independent tests, he will not be selected for the next round. But we also have a sentence revocation principle: anytime later, if contrary evidence is found, for a previously sentenced guilt individual by seemingly independent evidences, we need to reconsider the case, to find if there is a hidden common connection/factor of the original evidences; if there is, that common factor is the real culprit and the individual should be claimed innocent (and returned to the good citizen/candidate pool).

In an evolution process, we need an initial reason or pretense before the selection/action, but will re-explain or adjust our reason after the action, so the starting reasoning or principle is not actually so important and we think the evolution will bring us to the bright future (but if not, this is the history and so be it).

II. Relay Optimization Method

A. Dominance Property

In the VaR or TVaR constraint portfolio optimization problem, the candidate contracts contribute to the objective such as the expected profit, but at the same time compete for resources such as limit and risk limit. Even though they are not exactly additive, for example, "misaligned" contracts may not post noticeable additional risk limit; but to some extent we can consider them as exclusive: selecting more of one contract

F. X. Wang is with Validus Research Inc., Waterloo, ON N2J1R1 Canada (corresponding author phone: 519-783-9106; fax: 519-783-9101; e-mail: frank.wang@validusresearch.com).

may require us to select less of another contract. Numerical experiences indicate a lattice structure, and ranking or weak dominance properties among the contracts: there exist minimum and maximum elements. So we propose the following conjecture for our problem domain:

--If in all candidate solutions, a contract has solution limits very close to its minimum constraint, it means this contract is redundant, and its contribution can be replaced by other more resource-efficient contracts.

--If a contract has solution limits very close to its maximum constraint, it means this contract is one of the most resource-efficient and should be fully utilized.

If in multiple runs with randomly selected starting populations, a contract always has solution limits very close to zero, we believe it belongs to the first category, and vice versa. Based on this conjecture of some kind of "certainty out of randomness", a three-stage relay optimization method is designed and tested. Its effectiveness is checked against other methods and non-relay approaches.

B. A Real World Problem

We have a universe of 157 reinsurance contracts; their maximum limits, premium, and the loss matrix of loss scenarios by contracts are given.

The constraints are:

--Lower bound of the weights are 0 and upper bounds of the weights are 1.

-- Portfolio limit – portfolio premium = 1e8.

-- Portfolio 99% VaR <= 0.45 * portfolio limit.

-- Solution limit for each contract is either zero or above 1e6.

The objective is to maximize the expected profit of the portfolio (premium minus expected loss) or minimize the negative of the expected profit.

In many cases when the VaR constraint is not an in-effect constraint (far away from the boundary), it is in essence a small-sized linear program; there is no challenge at all. But when the VaR constraint is an in-effect constraint, it is a nonlinear and non-convex problem. Most algorithms will be trapped by a local optimum. The faster an algorithm converges, the more likely it will not arrive at the global optimum solution.

Essentially, we want to explore the solution space as completely as possible, and here we meet the challenge of dimension. Even if we only try the two points of minimum and maximum limit for each contract, there is a total of 1.82687704666363E+47 possible combinations.

If we can reduce the dimension from 157 to 17 for example, the combination will be reduced to 131,072, which seems more reachable. How can we reduce the dimension, and yet not lose the true solution? Our proposed conjecture leads us to a relay optimization method. Numerical experiment and comparison are conducted to confirm the conjecture and approach.

As a by-product, the exploration and exploitation performance of the evolved hybridized algorithm GA-MPC-OX are also established.

C. Three Stage Relay Optimization

We construct objective functions iteratively upon reducing the number of free-changing variables. For our example case, the first objective function has all the original 157 contracts (version obj_v1). The second objective function is obtained by setting all the contracts that in multiple runs of obj_v1 having solutions all equal to the minimum (or the maximum) limit to be fixed at that minimum (or maximum) limit: we get 130 free to change contracts(version obj_v1_1). And from solution to obj_v1_1, by carefully selecting the threshold to let contracts that are very close to the minimum or maximum limit to be fixed as the minimum or maximum limit, we get a third version of the objective function that has 17 free to change contracts (obj_v1_2).

Four algorithms GA_MPC [2], MOEAD [3], [4], ENSMOEAD [5], and DSA [6] proposed very recently are used to solve obj_v1. The best objective values we get are:

-- GA_MPC, 20 runs best at 15,311,378.

-- MOEAD, 20 runs best at 14,511,925.

-- ENSMOEAD, 20 runs best at 14,414,830.

-- DSA, 100,000 epochs run resulting in 14,861,177.

-- GA_MPC, another 20 runs best at 15,216,363.

GA_MPC is significantly better than other algorithms, so we use it in the other versions and get:

-- 20 runs for objective obj_v1_1 best at 15,392,950.

-- 20 runs for objective obj_v1_2 best at **15,399,491**.

A non-relay solution for obj_v1 seems impossible to get even close to the relay obj_v1_2 solution. For confirmation, sixteen algorithms: DyHF [7], CMODE [8], ICDE [9], PSO-DE [10], DSA [6], DECC-G [11], [12], CoDE [13], ETLBO [14], OXDE [15], MBA [16], IRM-MEDA [17], TLBO [14], MMEA [18], RM-MEDA [19], ABC [20], and IABC [21] are also used to solve obj_v_1_2. The resulting best objective values are in Table I. Among them, the MBA is the fastest algorithm, almost twenty times faster than all other algorithms. The CMODE is the most stable algorithm; almost over half the runs of it give results located in the higher quantiles of the objective value distribution from all the algorithms. But DyHF is the best algorithm, other than GA-MPC, to get the best objective value, in 20 runs.

GA-MPC has the best capability of arriving at an exceptionally good solution, but at the same time can fluctuate away from the good solution; we cannot make sure one set of 20 runs, 100 runs, or even 400 runs, will give the best values. The number from a stable algorithm such as CMODE can tell us when an exceptionally good solution is obtained.

Randomness is a double-edged sword. We need it in order to reduce the dimension. We also want the algorithm that can fluctuate towards the best solution. But at the same time, we also want to reach the best solution in as few as possible runs, with fewer fluctuations in the wrong direction.

To solve this perplex, we will apply the principle of hybridization, combine other algorithms with GA-MPC, and also adjust factors or strategies inside it, trying to find the real impacting factors through the mist or shadowing effect of the randomness, and modulate it (like applying the "mutation operator" to the candidate algorithm to get a team of children algorithms) so that we can have some certainty of getting the best solution in a reasonable number of parallel algorithm runs. Even though a champion may not be able to copy, a candidate team of past champions may have a higher chance of winning a future game, compared with a brute force method of continuing to run one algorithm.

TABLE I
ALGORITHMS AND BEST OBJECTIVE VALUES FOR OBJ_V_1_2

Algorithm	Best Objective Value
DyHF	15,397,037
CMODE	15,396,958
ICDE	15,395,697
PSO-DE	15,385,107
DSA	15,375,067
DECC-G	15,363,014
CoDE	15,360,104
ETLBO	15,341,005
OXDE	15,330,036
MBA	15,318,765
IRM-MEDA	15,300,754
TLBO	15,270,903
MMEA	15,264,939
RM-MEDA	15,209,221
ABC	15,038,804
IABC	14,148,533

Multiple runs of each algorithm are made while time permissible, ranging from several days to several weeks. For multiple objective optimization algorithms, the constraints' violations are summed to be the other objective; for single objective optimization algorithms, the constraints' violations are added to the objective value as penalty terms.

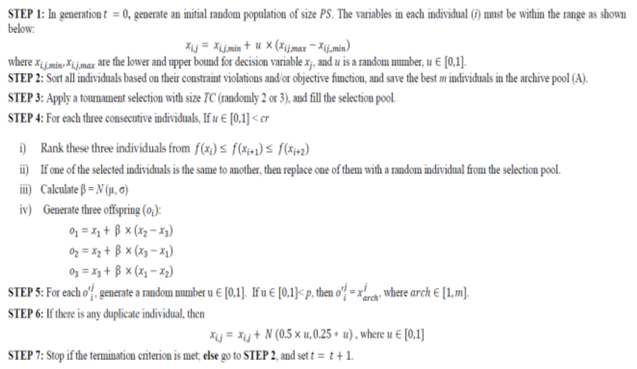
The rationale for this "team intelligence" is that a champion is possibly made purely by chance, but there should be a very small probability for a team of champions to achieve first place all by chance. This point of view will be tested numerically in section F.

D. Why GA-MPC

In our practice, several generations of algorithms are used. The first generation is the original genetic algorithm (GA) and global search algorithm such as pattern search algorithm. They can find better solutions than nonlinear constraint optimization algorithms. The second generation algorithm is PSwarm [22]; it outperformed the first generation algorithm and thus eliminated it. The third generation algorithm is CoDE, the best for our application among all differential evolution algorithm (DE), and better than PSwarm. Both of them are hybridized algorithms.

CoDE was later cast out by GA-MPC, an ingenious idea that "three parents generate three children through difference operators" by S. Elsayed, R. Sarker and D. Essam. This seems to be a simple and pure algorithm, in contradictory to our "hybridization principle". Actually, GA-MPC is also hybridization: in "spirit" genetic algorithm, but in format a variant of differential evolution, through the use of three difference operators, discrete counterparts of the gradient, with its effectiveness but without the need of calculating the gradient. This revolutionary combination classifies GA-MPC as a brand new strain from GA or DE, and better than both GA and DE.

The much simpler form of GA-MPC compared to other algorithms makes it possible to tune or experiment with its various ingredients, for achieving a repeatable good performance. The original GA-MPC algorithm [2] is listed below as a map for locating the places we are going to adjust:


STEP 1: In generation $t = 0$, generate an initial random population of size PS. The variables in each individual (i) must be within the range as shown below:

$$x_{i,j} = x_{i,j,min} + u \times (x_{ij,max} - x_{ij,min})$$

where $x_{i,j,min}, x_{i,j,max}$ are the lower and upper bound for decision variable x_j, and u is a random number, $u \in [0,1]$.

STEP 2: Sort all individuals based on their constraint violations and/or objective function, and save the best m individuals in the archive pool (A).

STEP 3: Apply a tournament selection with size TC (randomly 2 or 3), and fill the selection pool.

STEP 4: For each three consecutive individuals, If $u \in [0,1] < cr$

i) Rank these three individuals from $f(x_i) \leq f(x_{i+1}) \leq f(x_{i+2})$

ii) If one of the selected individuals is the same to another, then replace one of them with a random individual from the selection pool.

iii) Calculate $\beta = N(\mu, \sigma)$

iv) Generate three offspring (o_i):

$$o_1 = x_1 + \beta \times (x_2 - x_3)$$
$$o_2 = x_2 + \beta \times (x_3 - x_1)$$
$$o_3 = x_3 + \beta \times (x_1 - x_2)$$

STEP 5: For each o'^{j}_{i}, generate a random number $u \in [0,1]$. If $u \in [0,1] < p$, then $o'^{j}_{i} = x^{j}_{arch}$, where $arch \in [1, m]$.

STEP 6: If there is any duplicate individual, then

$$x_{i,j} = x_{i,j} + N(0.5 \times u, 0.25 * u)\text{ , where } u \in [0,1]$$

STEP 7: Stop if the termination criterion is met; **else** go to **STEP 2**, and set $t = t + 1$.

Fig. 1. GA with multi-parent crossover (GA-MPC) algorithm.

E. GA-MPC adjusted

The STEP 1 initial population selection can use low discrepancy sequence generators [23] [24], and can combine with the opposition based method [25], [26], the random number generator can use well equidistributed sequence algorithm [27] to improve its performance. But we are more interested in other factors that can show drastic effects when we change it.

The replication repair operator of STEP 6, as well as the boundary checking operator, is such a place. The normal distribution originally used tested poorly for our problem, instead, a threshold-based absorption in the boundary, or the same-threshold-based repair operator to replace the approximately repetitive individual with the boundary-trimmed new individual from the original individual, sees great improvement, if we use the least upper bound of the bounds that will absorb all the coordinates which appear to be small noises ("least upper bound principle"). This is like adding a boundary attraction potential, or an additional constraint of either at boundary or at least a threshold distance away.

For global optimization problems, form is important. There is also an observed "economical principle": add constraint in only when it is violated; even adding a not in effect constraint in will have effect, but adverse effect. Why do we see a "counterexample" here? This anti-intuition additional "constraint" which can lead to better solutions is perhaps because this guidance helps avoid vast barren lands of those small noises, and at the same time exactly matches our conjecture of the dominance.

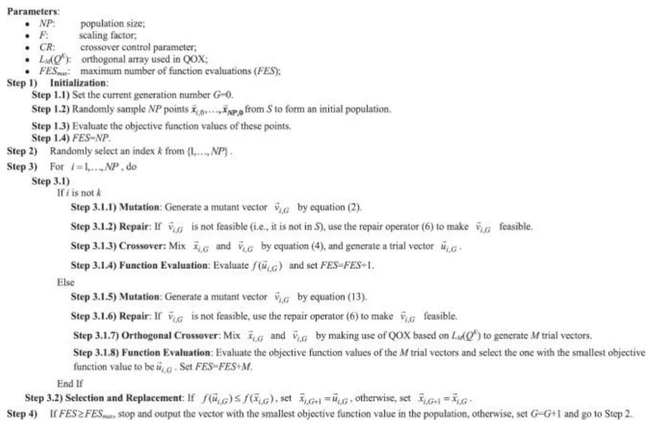
Another place is the crossover operator of STEP 4 iv). We will not follow its original reason of better or worse search direction but will use a geometric or symmetry reasoning. Numerical tests show when the three offspring use three different betas, we can get better effects than using a common beta. When we plot all the possible values of the beta together with the three differences vectors, we will see a counterclockwise-spinning fan.

So other shapes and their combinations are tested, among them are clockwise-spinning fan, arrived at by replacing the three differences vectors by their negative, arrow, windmill, hook, rays, combinations of different types of fans, etc.; the best shape is the clockwise-spinning fan, followed by the counterclockwise-spinning fan, and then the arrow.

A slowly-changing beta, only changing when several generations pass without objective value improvement, gives better results than changing it in every generation.

F. GA-MPC-OX

Professor Qingfu Zhang advised us that using orthogonal crossover (OX) may improve the search ability of an algorithm. In their paper [15] there is a detailed description of the orthogonal crossover algorithm:



Parameters:

- *NP*: population size;
- *F*: scaling factor;
- *CR*: crossover control parameter;
- $L_M(Q^K)$: orthogonal array used in QOX;
- FES_{max}: maximum number of function evaluations (*FES*);

```
Step 1) Initialization:
    Step 1.1) Set the current generation number G=0.
    Step 1.2) Randomly sample NP points $\vec{x}_{1,0},\ldots,\vec{x}_{NP,0}$ from S to form an initial population.
    Step 1.3) Evaluate the objective function values of these points.
    Step 1.4) FES=NP.
Step 2) Randomly select an index k from $\{1,\ldots,NP\}$.
Step 3) For $i=1,\ldots,NP$, do
    Step 3.1)
        If i is not k
            Step 3.1.1) Mutation: Generate a mutant vector $\vec{v}_{i,G}$ by equation (2).
            Step 3.1.2) Repair: If $\vec{v}_{i,G}$ is not feasible (i.e., it is not in S), use the repair operator (6) to make $\vec{v}_{i,G}$ feasible.
            Step 3.1.3) Crossover: Mix $\vec{x}_{i,G}$ and $\vec{v}_{i,G}$ by equation (4), and generate a trial vector $\vec{u}_{i,G}$.
            Step 3.1.4) Function Evaluation: Evaluate $f(\vec{u}_{i,G})$ and set FES=FES+1.
        Else
            Step 3.1.5) Mutation: Generate a mutant vector $\vec{v}_{i,G}$ by equation (13).
            Step 3.1.6) Repair: If $\vec{v}_{i,G}$ is not feasible, use the repair operator (6) to make $\vec{v}_{i,G}$ feasible.
            Step 3.1.7) Orthogonal Crossover: Mix $\vec{x}_{i,G}$ and $\vec{v}_{i,G}$ by making use of QOX based on $L_M(Q^K)$ to generate M trial vectors.
            Step 3.1.8) Function Evaluation: Evaluate the objective function values of the M trial vectors and select the one with the smallest objective
            function value to be $\vec{u}_{i,G}$. Set FES=FES+M.
        End If
    Step 3.2) Selection and Replacement: If $f(\vec{u}_{i,G}) \le f(\vec{x}_{i,G})$, set $\vec{x}_{i,G+1}=\vec{u}_{i,G}$, otherwise, set $\vec{x}_{i,G+1}=\vec{x}_{i,G}$.
Step 4) If $FES \ge FES_{max}$, stop and output the vector with the smallest objective function value in the population, otherwise, set G=G+1 and go to Step 2.
```

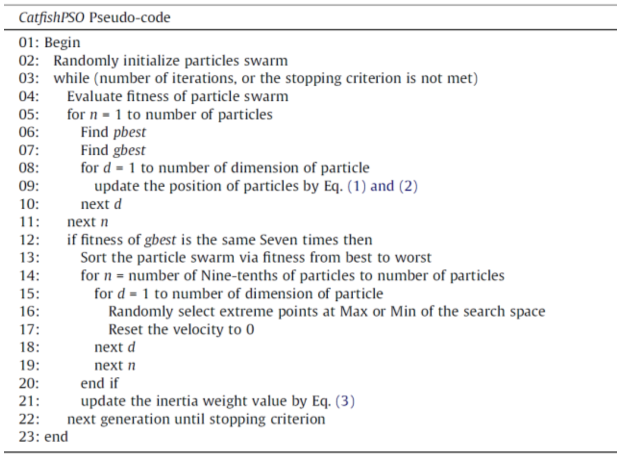
Fig. 2. Orthogonal crossover (OX) algorithm.

Since their paper shows that the combinations of 9, 25, 49, and 121 each has its merit for some test instances, we randomly select one from these four to use in each generation. For simplifying the implementation, instead of interpolation, we choose several elitist candidates, together with the randomly selected candidate, to form a set with the total number of the levels, sorting them and using them as a lookup table for the interpolated value.

The step 3.2) is a one-on-one competition for entering the selection. We compared the "replacing one individual only" method to "using all the orthogonal crossover generated offspring in the selection pool" method, and found that there is no significant difference, but sometimes the latter will get slightly better results, so we use the alternative method of using them all.

We combine this OX with GA-MPC (either before the mutation operator of STEP 5, or after it, with the previous method testing slightly better). Experiments show the combined algorithm GA-MPC with OX works as expected: it can find solutions with objective values that are almost not reachable by running the original algorithm with brute force for as many times as possible without OX.

To further combat the stagnant, the catfish algorithm [28] is also hybridized in (Fig 3). Their original method of replacing the 10% worst members tested worse than without using the catfish algorithm. We changed this to randomly select the 10% members to substitute in the lower half of the candidate pool. Tests showed improvement with this change.



CatfishPSO Pseudo-code

```
Begin
  Randomly initialize particles swarm
  while (number of iterations, or the stopping criterion is not met)
    Evaluate fitness of particle swarm
    for n = 1 to number of particles
      Find pbest
      Find gbest
      for d = 1 to number of dimension of particle
        update the position of particles by Eq. (1) and (2)
      next d
    next n
    if fitness of gbest is the same Seven times then
      Sort the particle swarm via fitness from best to worst
      for n = number of Nine-tenths of particles to number of particles
        for d = 1 to number of dimension of particle
          Randomly select extreme points at Max or Min of the search space
          Reset the velocity to 0
        next d
      next n
    end if
    update the inertia weight value by Eq. (3)
  next generation until stopping criterion
end
```

Fig. 3. Catfish algorithm.

Inspired by the exclusive property of our problem domain, we tested an alternative method to the catfish algorithm: instead of using minimum or maximum alternatively, we choose two coordinates randomly, increase one coordinate and decrease the other coordinate by two percentages selected from a normal distribution with a mean of 0.1 and standard deviation of 0.05. Tests show our "pivot" method is slightly better than the catfish method, so we use both of them, each with a probability of 0.5 of selection.

We call the thus hybridized GA-MPC, OX, catfish and pivot algorithm GA-MPC-OX. Now we have two adjustable factors: the stagnant generation numbers before engaging the catfish and pivot algorithm, which [28] took to be 7, and the stagnant generation numbers before changing the betas. Vast experiments show the following 11 combinations are good starting points: (25,5), (30,5), (19,9), (21,9), (28,9), (29,9), (30,9), (9,16), (29,16), (16, Inf), and (17, Inf). The top 17 objective values found through these GA-MPC-OX algorithms, as well as the top 4 algorithms of CEC'13 Special Session & Competition on Real-Parameter Single Objective Optimization [29] solving our obj_v1_2 are collected in table II.

These top algorithms tested worse than our customized GA-MPC-OX, and also worse than the not-adjusted algorithms DyHF and CMODE, perhaps because they are tuned for the benchmark problems, not for our problem.

People say we cannot enter the same river twice, but we think practice is the final test for truth. To also verify whether team intelligence can add certainty to our noisy environment, we reran the top 17 parameter combinations of GA-MPC-OX, and collected the results in table III, from which we can see that the "no twice" saying is exactly true, but inexactly false. One member replicated the best solution.

TABLE II
COMPARE GA-MPC-OX WITH OTHER ALGORITHMS VIA OBJ_V_1_2

Algorithm	Best Objective Value	Detailed description
GA-MPC-OX25	15,400,646	Clockwise spin difference operator, threshold (7,Inf)
GA-MPC-OX34-30-5	15,400,639	Clockwise spin difference

		operator, threshold (30,5)
GA-MPC-OX34-16	15,400,638	Clockwise spin difference operator, threshold (16,Inf)
GA-MPC-OX35-30-5	15,400,632	Change spin direction when change beta, threshold (30,5)
GA-MPC-OX34-9-16	15,400,625	Clockwise spin difference operator, threshold (9,16)
GA-MPC-OX34-30-9	15,400,624	Clockwise spin difference operator, threshold (30,9)
GA-MPC-OX6	15,400,616	Counterclockwise spin difference operator, threshold (Inf,Inf)
GA-MPC-OX31-19	15,400,613	Change spin direction when engage catfish or pivot, threshold (19,Inf)
GA-MPC-OX35-25-5	15,400,610	Change spin direction when change beta, threshold (25,5)
GA-MPC-OX34-29-16	15,400,422	Clockwise spin difference operator, threshold (29,16)
GA-MPC-OX34-21-9	15,400,420	Clockwise spin difference operator, threshold (21,9)
GA-MPC-OX34-17	15,400,417	Clockwise spin difference operator, threshold (17,Inf)
GA-MPC-OX32-21	15,400,415	Change spin direction each generation, threshold (21,Inf)
GA-MPC-OX34-28-9	15,400,400	Clockwise spin difference operator, threshold (28,9)
GA-MPC-OX34-29-9	15,400,393	Clockwise spin difference operator, threshold (29,9)
GA-MPC-OX19	15,400,392	Counterclockwise spin difference operator, threshold (7,Inf)
GA-MPC-OX1	15,400,250	Counterclockwise spin difference operator, threshold (Inf,Inf)
DyHF	15,399,842	100 runs used 175 hours
CMODE	15,396,996	100 runs used 172 hours
NBIPOPaCMA	15,396,794	1st run get best value by BIPOPaCMA
NBIPOPaCMA	15,388,441	2nd run get best value by NBIPOPaCMA
NBIPOPaCMA	15,390,048	3rd run get best value by NBIPOPaCMA
SHADE_CEC2013	15,393,402	45 run best
DRMA-LSCh-CMA	15,376,452	51 run best
iCMAES-ILS	15,208,803	20 run best

The pair of thresholds is for the stagnant generation numbers before engaging the catfish and pivot algorithm, and the stagnant generation numbers before changing the betas.

TABLE III
TOP 17 GA-MPC-OX PARAMETERS COMBINATIONS RERUN 20 TIMES

Algorithm	Best Objective Value	Detailed description
GA-MPC-OX25	15,397,159	Clockwise spin difference operator, threshold (7,Inf)
GA-MPC-OX34-30-5	15,396,357	Clockwise spin difference operator, threshold (30,5)
GA-MPC-OX34-16	15,396,974	Clockwise spin difference operator, threshold (16,Inf)
GA-MPC-OX35-30-5	15,397,227	Change spin direction when change beta, threshold (30,5)
GA-MPC-OX34-9-16	15,395,281	Clockwise spin difference operator, threshold (9,16)
GA-MPC-OX34-30-9	15,396,939	Clockwise spin difference operator, threshold (30,9)
GA-MPC-OX6	15,399,965	Counterclockwise spin difference operator, threshold (Inf,Inf)
GA-MPC-OX31-19	15,396,690	Change spin direction when engage catfish or pivot, threshold (19,Inf)
GA-MPC-OX35-25-5	15,397,019	Change spin direction when change beta, threshold (25,5)
GA-MPC-OX34-29-16	15,396,236	Clockwise spin difference operator, threshold (29,16)
GA-MPC-OX34-21-9	**15,400,644**	Clockwise spin difference operator, threshold (21,9)
GA-MPC-OX34-17	15,398,035	Clockwise spin difference operator, threshold (17,Inf)
GA-MPC-OX32-21	15,395,838	Change spin direction each generation, threshold (21,Inf)
GA-MPC-OX34-28-9	15,397,059	Clockwise spin difference operator, threshold (28,9)
GA-MPC-OX34-29-9	15,399,482	Clockwise spin difference operator, threshold (29,9)
GA-MPC-OX19	15,396,140	Counterclockwise spin difference operator, threshold (7,Inf)
GA-MPC-OX1	15,396,163	Counterclockwise spin difference operator, threshold (Inf,Inf)

The pair of thresholds is for the stagnant generation numbers before engaging the catfish and pivot algorithm, and the stagnant generation numbers before changing the betas.

G. *Further Test of GA-MPC-OX*

For the TVaR constrained portfolio optimization problem, there is a linearization method of transforming it to an equivalent linear problem by adding auxiliary variables [30], [31]. Using the then calculated accurate solution for the TVaR constrained problem as a benchmark, experiments show GA-MPC-OX will usually get the unique global solution in two to three runs. The first run is very close to the final solution, but with the running time of GA-MPC-OX reduced by 60% compared to the transformed linear program, both in Matlab implementation.

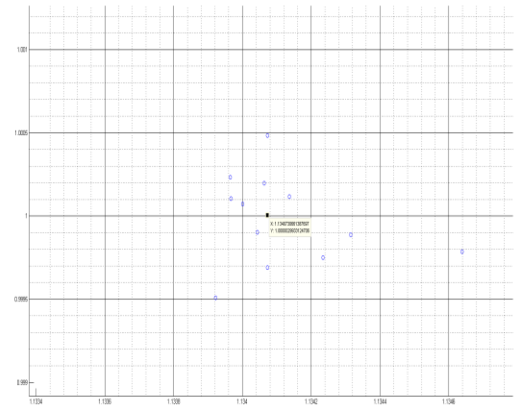


Fig. 4. VaR as a function of TVaR. The x-axis is the ratio of the used portfolio TVaR constraint to the given VaR constraint. The y-axis is the quotient of the calculated portfolio VaR divided by the given VaR constraint.

People usually use a TVaR approximation to solve the VaR constraint optimization problem, by solving a sequence of slightly larger TVaR constrained problems so that the solution will have calculated VaR approaching the given VaR constraint. The TVaR approximation method solution is shown in Fig 4 and Fig 5. The approximated expected profit is 15,329,291, which compared to our GA-MPC-OX solution is a very inaccurate "approximation".

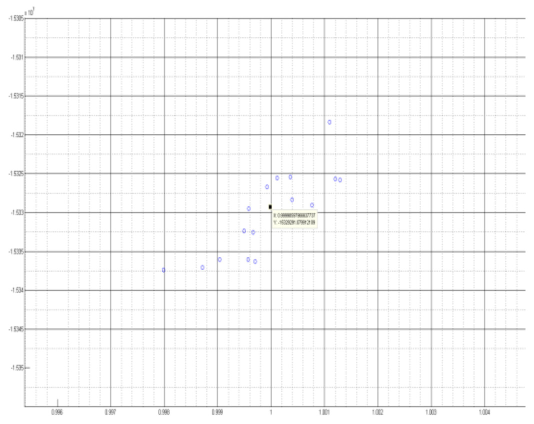


Fig. 5. Expected profit as an indirect function of TVaR. The x-axis is the solution portfolio calculated VaR divided by the given VaR constraint for the sequence of approximating TVaR. The y-axis is the solution portfolio negative expected profit.

There is some heuristic reason for this observation. From Fig 4, we can see that small changes of x values resulted in large fluctuations of y; this amounts to independence of x and y locally. In other words, when we fix the TVaR, the VaR can still change "arbitrarily", so the optimum solution will choose the maximum VaR, for return of biggest expected profit. So the resulting VaR is bloated for a given TVaR. When we want to limit the bloated VaR by the given VaR, the originally used TVaR will need to be smaller than it should be.

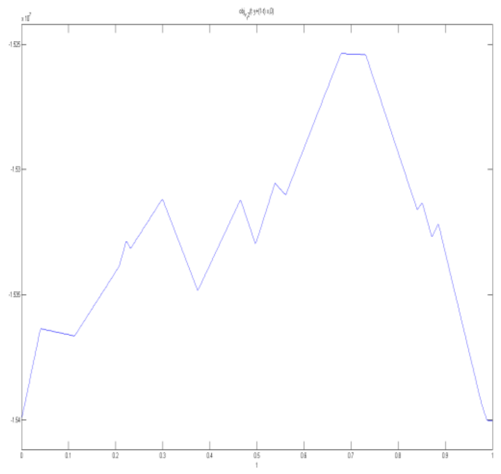


Fig. 6. How far away are two suboptimal basins? The x-axis is the linear interpolation between the best solution from GA-MPC that has an objective value of 15,399,490, and the 20 runs best solution from GA-MPC-OX1 that has an objective value of 15,400,250. The y-axis is the interpolated solution portfolio negative expected profit.

The best objective value GA-MPC can get is 15,399,490; what prevents it from getting numbers GA-MPC-OX can? From Fig. 6, we can see that the suboptimal solutions are narrow basins set apart by high mountains brought about by the VaR violation penalty, which cannot be trespassed directly. Another kind of solution differences are ones located in the same basin, there is a "feasible" line connecting them, but due to the high dimensionality, such a direction is hard to come by. This observation brings about the idea for our pivot algorithm.

TABLE IV
11 GA-MPC-OX PARAMETER PAIRS BEST OBJECTIVE VALUES FOR OBJ_V_1_2 PART I

Algorithm	Best Objective Value
GA-MPC-OX34-25-5	15,399,066
GA-MPC-OX34-30-5	15,396,357
GA-MPC-OX34-30-5	15,400,639
GA-MPC-OX34-19-9	15,399,068
GA-MPC-OX34-21-9	15,400,421
GA-MPC-OX34-21-9	15,400,644
GA-MPC-OX34-28-9	15,397,059
GA-MPC-OX34-28-9	15,400,400
GA-MPC-OX34-29-9	15,399,482
GA-MPC-OX34-29-9	15,400,393
GA-MPC-OX34-30-9	15,396,939
GA-MPC-OX34-30-9	15,400,624
GA-MPC-OX34-9-16	15,395,281
GA-MPC-OX34-9-16	15,400,625
GA-MPC-OX34-29-16	15,396,236
GA-MPC-OX34-29-16	15,400,422
GA-MPC-OX34-16-Inf	15,396,974
GA-MPC-OX34-16-Inf	15,400,638
GA-MPC-OX34-17-Inf	15,398,035
GA-MPC-OX34-17-Inf	15,400,417
GA-MPC-OX35-25-5	15,397,516
GA-MPC-OX35-25-5	15,400,610
GA-MPC-OX35-30-5	15,397,227
GA-MPC-OX35-30-5	15,400,632
GA-MPC-OX35-19-9	15,396,982
GA-MPC-OX35-21-9	15,396,703
GA-MPC-OX35-28-9	15,393,919
GA-MPC-OX35-29-9	15,396,494
GA-MPC-OX35-30-9	15,396,598
GA-MPC-OX35-9-16	15,396,831
GA-MPC-OX35-29-16	15,398,056
GA-MPC-OX35-16-Inf	15,395,312
GA-MPC-OX35-17-Inf	15,396,878
GA-MPC-OX36-25-5	15,397,162
GA-MPC-OX36-30-5	15,396,703
GA-MPC-OX36-19-9	15,395,882
GA-MPC-OX36-21-9	15,395,892
GA-MPC-OX36-28-9	15,395,338
GA-MPC-OX36-29-9	15,399,092
GA-MPC-OX36-30-9	15,400,416
GA-MPC-OX36-9-16	15,397,008
GA-MPC-OX36-29-16	15,400,421
GA-MPC-OX36-16-Inf	15,396,695
GA-MPC-OX36-17-Inf	15,396,713

These GA-MPC-OX algorithms use three parents to generate three children. Some parameter pairs have been run twice.

The 11 parameter combinations are tested by the clockwise spin search vector shape algorithm GA-MPC-OX34, switching between clockwise and counterclockwise spin search vector algorithm GA-MPC-OX35, counterclockwise spin search vector algorithm GA-MPC-OX36, as well as “four parents generating four children” clockwise spin search vector shape algorithm GA-MPC-OX38, and counterclockwise spin search vector algorithm GA-MPC-OX39. The tests showed some certainty of those parameters in getting good results, and also showed that the three parents method is better than the four parents method (Table IV & V), or maybe the four parents method has its special parameters.

TABLE V
11 GA-MPC-OX PARAMETER PAIRS BEST OBJECTIVE VALUES FOR OBJ_V_1_2 PART II

Algorithm	Best Objective Value
GA-MPC-OX38-25-5	15,399,484
GA-MPC-OX38-30-5	15,397,026
GA-MPC-OX38-19-9	15,397,063
GA-MPC-OX38-21-9	15,396,356
GA-MPC-OX38-28-9	15,396,941
GA-MPC-OX38-29-9	15,396,707
GA-MPC-OX38-30-9	15,397,033
GA-MPC-OX38-9-16	15,392,792
GA-MPC-OX38-29-16	15,397,025
GA-MPC-OX38-16-Inf	15,399,220
GA-MPC-OX38-17-Inf	15,396,490
GA-MPC-OX39-25-5	15,397,016
GA-MPC-OX39-30-5	15,396,562
GA-MPC-OX39-19-9	15,396,976
GA-MPC-OX39-21-9	15,397,032
GA-MPC-OX39-28-9	15,397,031
GA-MPC-OX39-29-9	15,397,031
GA-MPC-OX39-30-9	15,396,879
GA-MPC-OX39-9-16	15,396,884
GA-MPC-OX39-29-16	15,397,019
GA-MPC-OX39-16-Inf	15,397,516
GA-MPC-OX39-17-Inf	15,400,396

These GA-MPC-OX algorithms use four parents to generate four children.

III. CONCLUSION

The relay optimization method and GA-MPC-OX algorithm are proposed and tested for the VaR constraint portfolio optimization problem. Some starting parameter pairs are suggested for improving the outcome certainty by using them all. Its validity is also tested by additional experiments and can be evolved by a “season total” scoring method as used in sports to credit, eliminate, or revoke their position through each new test.

Other than our hybridization and parameter adjustment method for creating the team of algorithms to use, new competitive algorithms can also join in, substitute in, or combine with GA-MPC to form future generation teams.

ACKNOWLEDGMENT

F.X.Wang thanks Professor Qingfu Zhang for suggesting OX algorithm and sending him the source code, Professor Ponnuthurai Nagaratnam Suganthan and Dr. Ali Sadollah for sending him their source code, and all the authors who posted their codes online. F.X.Wang also thanks Dr. Lixin Zeng for forming the application framework, designing the various constraint scenarios, checking the running results, and giving feedback of questions and suggestions.

REFERENCES

[1] Lixin Zeng, “Optimizing a portfolio of insurance-linked securities,” unpublished.

[2] S. M. Elsayed, R. A. Sarker, D. L. Essam, “GA with a new multi-parent crossover for solving IEEE-CEC2011 competition problems,” in *Proc. IEEE Congr. Evol. Comput. (CEC), 2011*, pp. 1034–1040.

[3] Q. Zhang and H. Li, “MOEA/D: A Multi-objective Evolutionary Algorithm Based on Decomposition,” *IEEE Trans. on Evolutionary Computation*, vol.11, no. 6, pp712-731 2007.

[4] Q. Zhang, W. Liu, and H. Li, “The Performance of a New Version of MOEA/D on CEC09 Unconstrained MOP Test Instances,” Working Report CES-491, School of CS & EE, University of Essex, 02/2009.

[5] S. Z. Zhao, P. N. Suganthan, and Q. Zhang, “MOEA/D with an Ensemble of Neighbourhood Sizes,” *IEEE Trans on Evolutionary Computation*, (TEC) 16(3):442-446 (2012).

[6] P. Civicioglu, "Transforming Geocentric Cartesian Coordinates to Geodetic Coordinates by Using Differential Search Algorithm," *Computers and Geosciences*, 46, 229-247, 2012. Available: http://www.pinarcivicioglu.com/ds.html

[7] Y. Wang and Z. Cai, “A dynamic hybrid framework for constrained evolutionary optimization,” in *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol.42, no.1,2012, pp. 203-217.

[8] Y. Wang and Z. Cai, “Combining multiobjective optimization with differential evolution to solve constrained optimization problems,” in *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 1, 2012, pp. 117-134.CMODE

[9] G. Jia, Y. Wang, Z. Cai, and Y. Jin, “An improved (μ+λ)-constrained differential evolution for constrained optimization,” in *Information Sciences*, vol. 222, 2013, pp. 302-322.ICDE

[10] H. Liu, Z. Cai, and Y. Wang, “Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization,” *Applied Soft Computing*, vol. 10, no. 2, pp. 629-640, 2010.PSO-DE

[11] Zhenyu Yang, Ke Tang and Xin Yao, "Large Scale Evolutionary Optimization Using Cooperative Coevolution," *Information Sciences*, 178(15):2985-2999, 2008.DECC-G

[12] Zhenyu Yang, Ke Tang and Xin Yao, "Self-adaptive Differential Evolution with Neighborhood Search," in *Proceedings of the 2008 IEEE Congress on Evolutionary Computation (CEC2008)*, Hongkong, China, 2008. SaNSDE

[13] Y. Wang, Z. Cai, and Q. Zhang, “Differential evolution with composite trial vector generation strategies and control parameters,” *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55-66, 2011.CoDE

[14] R. V. Rao, V. Patel, “An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems,” *International Journal of Industrial Engineering Computations*, Volume 3, Issue 4, 2012, 535-560. Available: https://sites.google.com/site/tlborao/

[15] Y. Wang, Z. Cai, and Q. Zhang, “Enhancing the search ability of differential evolution through orthogonal crossover,” *Information Sciences*, vol. 185, no. 1, pp. 153-177, 2012.OXDE

[16] Ali Sadollah, Ardeshir Bahreininejad, Hadi Eskandar, and Mohd Hamdi, “Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems,” *Applied Soft Computing*, Volume 13, Issue 5, May 2013, pp2592–2612.MBA

[17] Y. Wang, J. Xiang, and Z. Cai, “A regularity model-based multiobjective estimation of distribution algorithm with reducing

redundant cluster operator," *Applied Soft Computing*, vol. 12, no. 11, pp. 3526-3538, 2012.IRM-MEDA

[18] A. Zhou, Q. Zhang and Y. Jin, "Approximating the Set of Pareto Optimal Solutions in Both the Decision and Objective Spaces by an Estimation of Distribution Algorithm," *IEEE Trans on Evolutionary Computation*, vol. 13, no. 5, pp1167-1189, 2009.MMEA

[19] Qingfu Zhang, Aimin Zhou and Yaochu Jin, "RM-MEDA: A Regularity Model Based Multiobjective Estimation of Distribution Algorithm," in *IEEE Trans. on Evolutionary Computation*, vol. 12, no. 1, pp41-63, 2008.

[20] D. Karaboga, B. Basturk, "A powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm," *Journal of Global Optimization*, Volume:39, Issue:3,pp:459-171, November 2007. Available: http://www.mathworks.com/matlabcentral/fileexchange/27125-solution-to-economic-dispatch-by-artificial-bee-colony-algorithm/content/ABC-eld/runABC.m

[21] P. W. Tsai, J. S. Pan, B. Y. Liao, and S. C. Chu, "Enhanced artificial bee colony optimization," *International Journal of Innovative Computing, Information and Control*, vol. 5, no. 12, pp. 5081–5092, 2009. IABC

[22] A. I. F. Vaz and L. N. Vicente, "A particle swarm pattern search method for bound constrained global optimization," *Journal of Global Optimization*, 39 (2007) 197-219. Available: http://www.norg.uminho.pt/aivaz/pswarm/

[23] S. Joe, F. Y. Kuo, "Remark on Algorithm 659: Implementing Sobol's quasirandom sequence generator," *ACM Trans. Math. Softw.* 29, 49-57 (2003).

[24] S. Joe, F. Y. Kuo, "Constructing Sobol sequences with better two-dimensional projections," *SIAM J. Sci. Comput.* 30, 2635-2654 (2008).

[25] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Trans. Evol. Comput*., vol.12, pp.64 -79 2008.

[26] S. Das and P. N. Suganthan, "Differential Evolution: A Survey of the State-of-the-art," *IEEE Trans. on Evolutionary Computation*, Vol. 15, No. 1, pp. 4-31, Feb. 2011.

[27] François Panneton, Pierre L'Ecuyer, Makoto Matsumoto, "Improved long-period generators based on linear recurrences modulo 2," *ACM Transactions on Mathematical Software (TOMS)*, v.32 n.1, p.1-16, March 2006. Available: http://www.iro.umontreal.ca/~panneton/WELLRNG.html

[28] Li-Yeh Chuang, Sheng-Wei Tsai, Cheng-Hong Yang, "Chaotic catfish particle swarm optimization for solving global numerical optimization problems," Applied *Mathematics and Computation*, 217 (2011) pp6900–6916.

[29] Special Session & Competition on Real-Parameter Single Objective Optimization at CEC-2013, Cancun, Mexico 21-23 June 2013. [Online]. Available: http://www.ntu.edu.sg/home/EPNSugan/index_files/CEC2013/CEC2013.htm

[30] Pavlo Krokhmal, Jonas Palmquist, and Stanislav Uryasev, "Portfolio optimization with conditional value-at-risk objective and constraints," *Journal of Risk*, 4 (2002): 43-68.

[31] Xiaodi Bai, Jie Sun, Xiaoling Sun, Xiaojin Zheng, "An Alternating Direction Method for Chance-Constrained Optimization Problems with Discrete Distributions," November 2013. Available: http://www.optimization-online.org/DB_FILE/2012/04/3448.pdf