A Population-based Incremental Learning Method for Constrained Portfolio Optimisation

Yan Jin, Rong Qu, Jason Atkin
The University of Nottingham
School of Computer Science
ASAP Group,
NG8 1BB, UK
Email: {ywj,rxq,jaa}@cs.nott.ac.uk

***Abstract*—This paper investigates a hybrid algorithm which utilizes exact and heuristic methods to optimise asset selection and capital allocation in portfolio optimisation. The proposed method is composed of a customised population based incremental learning procedure and a mathematical programming application. It is based on the standard Markowitz model with additional practical constraints such as cardinality on the number of assets and quantity of the allocated capital. Computational experiments have been conducted and analysis has demonstrated the performance and effectiveness of the proposed approach.**

I. Introduction

PORTFOLIO optimisation (PO) is one of the core areas in the financial study of making investment decisions. Since the seminal work by Markowitz in 1952 [1], this line of research has developed dramatically from both the theoretical and application aspects. Various linear programming (LP) solvable PO models proposed by scholars are discussed in [2]. A good review of the applications of metaheuristics to the PO problem is available in [3].

In this paper, we use the Markowitz Mean-Variance (MV) model for the PO problem. It is based on a single period of investment in a market with N different assets. Each asset i is associated with an expected return r_i and has a covariance σ_{ij} with each other asset j. The continuous decision variables x_i which represent the proportion of the capital that the investor allocates to each of the various assets available are assigned non-negative values (short sales are not allowed) and all x_i sum to 1 (the whole capital should be invested). A portfolio consists of a set of assets selected. The aim is to find a trade-off between maximising the total return r_P while minimising the risk exposure σ_P of the portfolio. The formulation of the model can be stated as follows:

Minimise

$$\lambda \cdot \sigma_P - (1-\lambda) \cdot r_P \tag{1}$$

Subject to

$$r_P = \sum_{i=1}^{N} x_i r_i \tag{2}$$

$$\sigma_P = \sum_{i=1}^{N} \sum_{j=1}^{N} \sigma_{ij} x_i x_j \tag{3}$$

$$\sum_{i=1}^{N} x_i = 1 \tag{4}$$

$$0 \leqslant x_i \leqslant 1, i = 1, ..., N \tag{5}$$

Where a parameter λ $(0 \leq \lambda \leq 1)$ is introduced to define the balance between the two objectives. A set of risk-return points can be obtained by solving the above model repeatedly, with different values of λ to investigate the trade-off between risk and return. For each point, there should be no portfolio with a higher expected return at the same risk level, and no portfolio with a lower risk at the same level of return. The set of points form the efficient frontier (EF).

Despite the theoretical value of this model, some significant drawbacks have been identified following many studies of the MV model. Firstly, it is arguable that it assumes that return is normally distributed [4]. In [3], the authors pointed out that gathering enough data to estimate risk and return might itself be difficult. Moreover, measurement errors can have a big effect on the estimation of risk and return from historical data [5]. As a result, many extensions of the original MV model have been introduced. These include the adoption of some alternative risk measures like mean-absolute deviation [6] or semi-absolute deviation [7]. Robust optimisation models deal with uncertainty sets of the mean and covariance of return in the MV model. Goldfarb and Iyengar [8] proposed a robust factor model for the returns and reformulated the uncertainty set to a second-order cone program (SOCP) which can then be solved efficiently.

Additionally, from a practical point of view, the MV model does not incorporate real-world conditions. It tends to suggest that investors hold as many assets as possible in order to diversify the portfolio and thus reduce the risk. However, according to [9], investors prefer a small set of assets as the administration of a large portfolio is prohibitively complex.

In this paper, we will mainly focus on the constrained problem, where quantity and cardinality constraints have been added to extend the original MV model. The quantity constraint defines the upper (ε) and lower (δ) bounds for the allowed investment for each asset in a portfolio. The cardinality constraint requires that there are only a limited number of assets in the portfolio. This constraint is modelled by introducing a binary variable z_i into the model, which has the value 1 if the asset is selected and 0 otherwise. The new constraints can be formulated as follows, resulting into a new model for the mixed integer problem:

$$\sum_{i=1}^{N} z_i = K \tag{6}$$

$$\varepsilon z_i \leqslant x_i \leqslant \delta z_i, i = 1, ..., N \tag{7}$$

$$z_i \in \{0, 1\}, i = 1, ..., N \tag{8}$$

The original MV model is normally represented in the form of minimizing the risk ($\lambda = 1$) with an additional constraint $r_P \geqslant Return^*$, where $Return^*$ is a lower bound of the expected return. This is a standard quadratic programming (QP) problem and has been extensively studied using exact methods, such as branch&bound or simplex methods to efficiently produce optimal solutions for small and medium sized instances. Alternatively, for maximising the return ($\lambda = 0$) with a maximum risk value, $Risk^*$, which can be accepted, the constraint $\sigma_P \leqslant Risk^*$ is added and the PO problem becomes a quadratically constrained program (QCP). This can be cast as a second order cone programming (SOCP) problem and then solved efficiently using methods like the interior point method, which was proposed in the last few years [10]. The conic programming techniques have proved quite efficient in solving non-linear convex optimisation problems and SOCP has proved effective for linear problems and special cases of quadratic problems. However, when considering large problem instances with practical constraints such as the cardinality constraint, Eq.(6), there currently exists no efficient algorithm in the literature [11], [12]. Therefore many applications resort to heuristic optimisation techniques for solving such constrained models.

The advantage of heuristic methods is that they can deal with complex constraints more easily and generate competitive solutions at a reasonable computational cost. Chang et al. [13] presented three heuristic algorithms based on genetic algorithms, tabu search and simulated annealing, which are able to find the estimated efficient frontiers while considering cardinality and quantity constraints. The computational applications of Genetic Algorithms in particular for solving PO problems were discussed in [14]. Some nature-inspired metaheuristics, such as swarm intelligence algorithms including ant colony optimisation [15] and particle swarm optimisation [12] have already demonstrated good performance. However, heuristics do not guarantee to find an optimal solution and performance is not always stable.

In recent years, hybrid techniques also showed promising results for solving the PO problem, particularly hybrids of exact and heuristic methods, which are also called matheuristics. A hybrid metaheuristic was developed for the constrained problem [16], which uses a local search to select the assets in the portfolio. At each step a quadratic procedure is implemented to allocate the best weights for the chosen assets. Similarly, a hybrid scheme which combines ideas from metaheuristics and quadratic programming provided a good and efficient solution to the MV model with several practical constraints [17]. It is shown that hybrid approach can provide competitive results in terms of accuracy and efficiency compared to the traditional metaheuristics like [13].

In this work, we derive an approach that hybridises different mathematical programming strategies with a Population-based Incremental Learning algorithm (PBIL), which is one of the estimation of distribution algorithms (EDA) and evolutionary processes. In order to obtain a comprehensive understanding of the proposed algorithm, we also evaluate a similar metaheuristic technique called the Compact Genetic Algorithm (CGA), which replaces the PBIL component in the hybrid algorithm. The results show that the PBIL based approach is more powerful in exploring the solution space and obtains better results.

The remainder of this paper is structured as follows. In Section 2, a detailed description of the solution procedure is provided, including our hybrid approach and constraint handling procedure. Section 3 reports datasets, parameter settings and computational results. Finally, this paper concludes in Section 4 with a summary and future work.

II. The Proposed Framework

In this paper, our hybrid strategy is based on a population-based metaheuristic and an exact method which iteratively generates new solution populations according to an updated real-valued vector of probabilities. The goal is to take the strength of both techniques to solve the constrained PO problem. The proposed method consists of two components. The first component deals with the discrete optimisation problem, choosing a set of assets, using the metaheuristic technique. The second component uses an exact mathematical technique to solve the continuous optimisation problem, with the discrete variables being fixed, i.e. to allocate weights to the assets which have been selected by the first component.

The metaheuristics that we investigate are Population-based Incremental Learning (PBIL) and the Compact Genetic Algorithm (CGA). Both of these employ a probability vector to represent the probability distribution of the population, which is updated through an evolutionary process. We use different successive quadratic programming (sQP) techniques within CPLEX, based on different problem models. The external commercial solver is embedded within the PBIL (or CGA) process, assigning values to the continuous variables and finding the fitness for each candidate solution.

A. Overview of underlying metaheuristics

1) Population-based incremental learning: PBIL was first introduced by Shumeet Baluja in 1994 [18] and was considered as an improvement of the genetic algorithm (GA). In PBIL, solutions in the population are encoded as a binary string of fixed length. Unlike GA, it does not apply operators like crossover and mutation to individuals in the population, but adopts the evolutionary process to acquire knowledge of the global optimum in the solution space. It uses a vector of real valued probabilities (the Probability Vector, PV), which reflects statistical information about the distribution of good individuals during the evolution, to generate high quality solutions with high probabilities when sampled.

Initially, all of the values of the PV are set to 0.5. Sampling from this vector yields random binary solution vectors because the probability of generating a 1 or 0 is equal. As the evolution progresses, the values in the PV gradually shift to represent higher quality solution vectors. This is accomplished by updating the PV based on the elite solutions from the generated solution vector. The scale by which the probability vector is updated is defined by a learning rate parameter (LR). The PV is also mutated with a specified mutation rate on a random basis. In the next generation, a new set of solution vectors is produced by sampling from the updated probability vector, and the cycle is continued until a stopping condition is met.

As the search progresses, entries in the probability vector move away from their initial settings of 0.5 towards either 0.0 or 1.0. The probability vector can be viewed as a prototype vector which stores knowledge of the search space during the evolution to generate better solution vectors over generations.

2) Compact Genetic Algorithm: As another estimation of distribution algorithm, CGA was firstly introduced by Harik et al. in 1999 [19]. Similarly to PBIL, its population is represented by a probability vector. However, at each generation it only generates two individual solutions sampled from the PV and only one tournament is performed between them. The PV is then adjusted and shifted towards the better one with a step parameter (n). The general procedure for a standard CGA is almost the same as the PBIL except for two factors: the sample size and the PV update scheme. It is worth noting that there is no mutation process for the PV in the CGA.

B. Individual representation and fitness evaluation

In our hybrid procedure, each solution is associated with both a binary vector and a real-valued vector. The length of each solution vector is based on the dimension of the problem instance (the total number of assets). Each element of the vector represents an available asset. The binary vector determines the value of z_i, which denotes whether an asset has been selected by the metaheuristic. The real-valued vector represents the weights allocated to the selected assets (x_i) and is dealt with by relevant mathematical programming optimizers in CPLEX. The choice of optimizer used in CPLEX depends upon the model we choose. In the case of $\lambda = 0$, the model is a QCP (cast as a SOCP) and can be solved by the Barrier optimizer. On the other hand, when $\lambda = 1$, the model is QP and can be solved by several available optimizers (e.g. Primal, Dual or Barrier). Each solution in the population is evaluated using the fitness value calculated by the solver. At each iteration of PBIL (or CGA), the PV is updated based on the best solutions obtained and is then used to generate an improved population in the next generation.

Algorithm 1 Hybrid Algorithm

```
1:  BEGIN
2:  Initialise Probability Vector: $PV[i] \leftarrow 0.5, i = 1, ..., N$
3:  repeat
4:      // Generate samples of size S from PV[i]
5:      for each j, $j = 1, .., S$ do
6:          for each i, $i = 1, ..., N$ do
7:              $binary\_vectors[j][i] \leftarrow$ generate sample binary variables based on $PV[i]$
8:          end for
9:          CardinalityControl(binary_vectors [j][i]) //See Algorithm 2
10:     end for

11:     // Evaluate samples
12:     Create SOCP (or QP) models for each of the $binary\_vectors[j][i]$
13:     Solve the models using CPLEX solver

14:     // Update Probability Vector towards elite solutions
15:     LearnPV($PV[i]$) //See Algorithm 3

16:     // Mutation process only in PBIL
17:     $r \leftarrow random(0.0, 1.0)$ // pm: mutation probability
18:     if $r < pm$ then
19:         Mutation ($PV[i]$) //See Algorithm 4
20:     end if
21: until Termination condition is met
22: END
```

C. The overall procedure of the proposed hybrid approach

The hybrid approach described in Algorithm 1 deals with two sub-problems, namely asset selection and weight allocation. Two solvers are involved, a master solver and a slave solver. The master solver is the PBIL (or CGA) and the exact technique (QP or SOCP) is the slave solver that acts as an assistance tool for the master solver. The master procedure randomly generates sample binary solutions which decide upon the assets to be included in the portfolio according to the values in the probability vector $PV[i]$. The binary solutions then serve as inputs for the mathematical models which determine the precise weights for these selected assets. The fitness values for each candidate portfolio solution are sent back to the master solver as feedback to update $PV[i]$. In each generation, $PV[i]$ is updated towards elite elements with better fitness. This process is repeated until a termination condition is met.

As described above, each candidate solution is adjusted in a 2N dimensional search space after each iteration, and each candidate must be feasible and satisfy Eqs.(6)-(7). Since the proportion constraint Eq.(7) can be easily modelled as a typical convex problem and solved efficiently, we can directly formulate it into the mathematical model, dealt with by the

Algorithm 2 CardinalityControl(binary_vectors [j][i])

```
1: while $Sum(binary\_vectors[j][i]) > K$ do
2:    among those entries $i$ in $binary\_vectors[j][i]$ with value 1
3:    $r = random(0.0, 1.0)$
4:    if $r > 0.5$ then
5:       set the entry with lowest PV[i] to 0
6:    else
7:       randomly select an entry and set it to 0
8:    end if
9: end while
10: while $Sum(binary\_vectors[j][i]) < K$ do
11:    among those entries $i$ in $binary\_vectors[j][i]$ with value 0
12:    $r = random(0.0, 1.0)$
13:    if $r > 0.5$ then
14:       set the entry with the highest PV[i] to 1
15:    else
16:       randomly choose one entry and set it to 1
17:    end if
18: end while
```

Algorithm 3 LearnPV (PV[i])

```
1: Case 1: in PBIL
2: for each $j, j = 1, .., Elite$ do
3:    $PV[i] \leftarrow PV[i] * (1.0 - LR) + binary\_vectors[j][i] * LR$
4: end for

5: Case 2: in CGA
6: if $f\_winner[i] \neq f\_loser[i]$ then
7:    if $f\_winner[i] = 1$ then
8:       $PV[i] \leftarrow PV[i] + 1/n$
9:    else
10:      $PV[i] \leftarrow PV[i] - 1/n$
11:   end if
12: end if
```

Algorithm 4 Mutation (P[i])

```
1: if $PV[i] > 0.5$ then
2:    $PV[i] \leftarrow PV[i] * (1.0 - ms)$
3: else if $PV[i] < 0.5$ then
4:    $PV[i] \leftarrow PV[i] * (1.0 - ms) + ms$
5: end if
```

solver. However, the cardinality constraint Eq.(6) makes the problem non-convex and increases the complexity of the problem for the exact solvers. PBIL and CGA are good at dealing with discrete variables and are therefore employed.

To explain the handling process for the cardinality constraint further, suppose $K^* = Sum(binary\_vectors[j][i])$, which represents the number of distinct assets i held by a portfolio j. If $K^* > K$, some assets should be removed; while if $K^* < K$, some assets should be added. This study suggests two arrangements with equal probabilities for both situations. One is randomly removing or selecting assets and another is keeping or selecting the K assets with the highest PV values. The details of the process are presented in Algorithm 2. The update scheme for the $PV[i]$ values in PBIL and CGA is described in Algorithm 3. In PBIL, PV is updated towards the best (elite) solutions, the number of which is denoted by Elite. In CGA, since only two solutions are sampled in each generation, f_winner and f_loser denote the better and worse solution respectively. The PV is updated by an amount specified by the step value, 1/n, where n is an integer. In addition, the mutation procedure which is used by the PBIL is described in Algorithm 4. The parameter ms controls the amount a mutation alters the value at each bit position.

D. Dynamic Adjustment Strategy

In order to improve the efficiency of the proposed hybrid approach, the following dynamic adjustment strategies are proposed:

1) Adjustment of the population size:
   As the algorithm progresses, the probability vector (PV) used tends to converge through evolution, which means a few good binary vectors have a much higher probability to be generated in each generation.
   Therefore in order to maintain the diversity and improve the efficiency, the duplicated candidate binary vectors are eliminated. This reduces the total amount of work to be processed by the slave solver (CPLEX), thus increasing the total efficiency significantly.
2) Adjustment of the search space and stopping condition: In the early generations, it is not necessary for CPLEX to thoroughly explore solutions for many candidate binary solutions which are not good enough. In order to not spend unnecessary effort, the search activity of the slave solver for each candidate binary solution is limited to a short time (10s) to quickly produce a feasible solution for updating PV. In the later generations, good binary solutions tend to be generated anyway which normally require little time for CPLEX to compute good solutions. The stopping condition of the whole hybrid approach is when either of the following conditions is met:
   a) the maximum number of iterations is reached
   b) the best fitness value so far has not changed for a fixed number of generations

Together these enhancements decrease the search time and prevent needlessly re-evaluating solutions and significantly improve the algorithm effectiveness.

III. Empirical Evaluation

A. Benchmark datasets and performance measures

The datasets tested in this paper are based on 5 well-known indices, the Hong Kong HangSeng, the German DAX100, the UK FTSE100, the US S&P100 and the Japan Nikkei, with dimensions of 31, 85, 89, 98 and 225 assets, respectively. These datasets were extracted by Beasley [20] based on weekly price data from March 1992 to September 1997, and are available in the OR-Library repository at http://people.brunel.ac.uk/~mastjjb/jeb/info.html. They have been widely tested in the literature for the PO problem.

We first take the results for the unconstrained PO problem from OR-Library to compute the unconstrained efficient frontier (UEF), determining upper bounds for the return with fixed risks for the constrained PO problem. The frontiers obtained under the quantity and cardinality constraints by our proposed hybrid approach are considered as an approximate efficient frontier (AEF). To evaluate the effectiveness and efficiency of the proposed approach, three performance measures are adopted, for each of which the smaller values denote better solutions.

The Average Percentage Error (APE) [16] measures the deviation of the obtained AEF from the UEF. It can be calculated using Formula (8), where x^* and x denote the points on the AEF and UEF, respectively, and f_i^r is the function value of the expected return (or risk), $i = 1...p$, where p is the number of points on the frontier.

$$\frac{100}{p} * \sum_{i=1}^{p} \frac{f_i^r(x^*) - f_i^r(x)}{f_i^r(x)} \tag{8}$$

Mean Euclidean Distance (MED) [12] refers to the average minimum distance of each point on the obtained AEF from the UEF, calculated using Formula (9). f_v and f_r are the risk (using Eq.(3)) and return (using Eq.(2)) of the portfolio. This measure presents the diversity and convergence of the obtained solutions.

$$MED = \frac{1}{p} * \sum_{x^* \in X_{AEF}} min\{d_{x^*x} | x \in X_{UEF}\} \tag{9}$$

$$d_{x^*x} = \sqrt{(f_r(x^*) - f_r(x))^2 + (f_v(x^*) - f_v(x))^2} \tag{10}$$

Algorithm Effort (AE) [21], which can be formulated as $AE = \frac{Time}{p}$, measures the ratio of the total run time to the number of feasible points obtained on the AEF.

B. Parameter and implementation setting

In the constrained instances, we set $\varepsilon = 0.01$, $\delta = 1$ as the lower and upper bounds of the weights, and $K = 10$ for the cardinality constraint. These have been widely used in the literature. The number of sample solutions in PBIL is $S = 100$ and the elites to be learned from is $Elite = 2$. The remaining parameters are set as follows: $LR = 0.25$, $pm = 0.2$, $ms = 0.05$, $n = 20$, $update\_Prob = 0.5$. These values are selected based on preliminary experimental results. All of the experiments conducted were coded in C++ in Microsoft Visual Studio 2010, linked with CPLEX 12.6 (64-bit) and were executed on a PC with Windows 7 (64-bit) Operating System, 6GB of RAM, and an Intel Core i7 CPU (960@3.2GHz).

C. Solver's Performance

According to the release notes for CPLEX 12.6, one of its newly added core features is for solving nonconvex problems. CPLEX 12.6 is capable of solving a quadratic program to global optimality and no longer requires the convexity of the problem. In order to understand and evaluate our algorithm more comprehensively, it is worth reporting the performance of CPLEX 12.6 on the proportion (C1) and cardinality (C2) constrained PO problem.

We tested stand-alone CPLEX to solve the constrained PO problems ($\varepsilon = 0.01$, $\delta = 1$, $K = 10$) on all of the datasets, to evaluate its performance. Solvers for both the problem of minimising risk (QP) and maximising return (QCP) were tested and for each case, CPLEX dynamically chose the suitable optimizer. We chose 50 points with equally spaced values across the EF from the OR-Library and set the time limit to 200s for each point.

In the experimental results, firstly for C1 only, both models were solved efficiently by CPLEX, obtaining globally optimal solutions for all of the sampled points in all of the datasets. The QP model took much less time than the QCP model. When both C1 and C2 were applied to the problem, the model became a mixed integer QP or QCP, for which CPLEX can obtain satisfactory results for many of the cases. Results are shown in Figure 1, where the rate of Optimal, Feasible and Infeasible solutions is presented, indicating that the solution status was proven to be optimal, feasible or infeasible respectively. Feasible means that only a feasible rather than provable optimal solution was obtained within the time limit, and infeasible means that there was no feasible solution at the level of return (or risk) value. It can be seen that for most of the points, optimal solutions can be obtained within a reasonable time, however, this did not work for all cases. We further tested those points where only feasible solutions could be obtained, without imposing a time limit. A significant computational time was required in order to find the optimal solutions for some of these cases. In particular, for N98, some of the points required more than 30000s to reach the optimal solution. This is due to the equality constraint of integer which increases the complexity based on the original MV model. In addition the solver for the QP model performed better than that for QCP in terms of both solution quality and efficiency.

In general, the results from the pure solver indicate that the exact methods which exist today are both powerful and suitable for solving the unconstrained problems with only the C1 constraint applied, however C2 considerably increases the complexity to generate feasible and good solutions for some instances. This encourages studies to resort to heuristic solution for these complex practical constraints. Moreover, the QP formulation is more suitable than the QCP for this problem, which suggests that we adopt QP in our hybrid procedure.

D. Proposed Algorithm for Constrained PO

In this section, we present the computational results of the hybrid approach for the constrained PO problem where the QP formulation was chosen due to its performance in CPLEX presented above. To produce the AEF, multiple runs were required, each with a prescribed return level. The return values

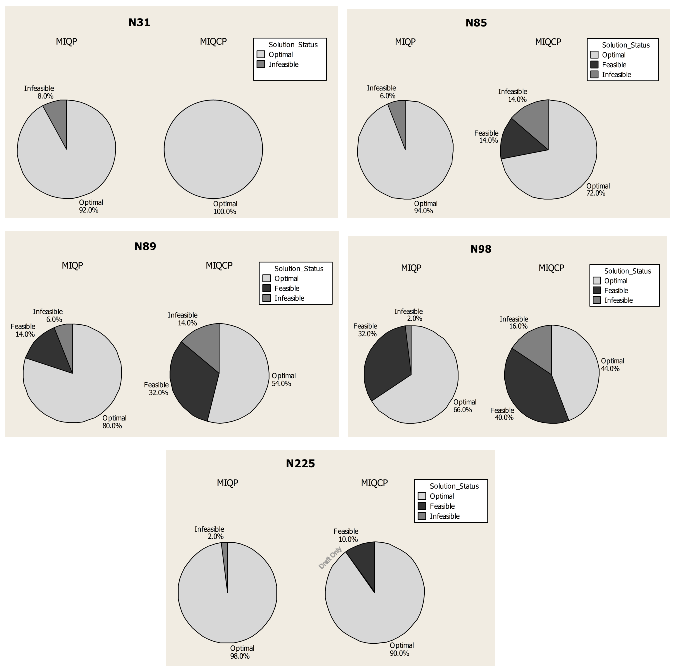

Fig. 1. Performance of the stand-alone CPLEX Solvers

TABLE I
AVERAGE TIME (S) FOR EACH POINT USING CPLEX

		N31	N85	N89	N98	N225
QP	C1	0.035	0.113	0.127	0.174	0.365
	C1&C2	0.031	5.080	41.294	68.719	0.591
SOCP	C1	0.2255	3.5996	7.0106	10.6957	5.6861
	C1&C2	0.102	36.367	75.402	96.781	32.849

chosen were evenly distributed on the UEF, and a total of 50 points were obtained. Each procedure was run 5 times with a different random number seed to obtain averaged results. The termination condition was when either that the total time reaches 200s or the best fitness did not change for 50 iterations. The time limit for the CPLEX solver for each single evaluation is set to 10s. The results for our proposed hybrid of PBIL and QP solver (PBIL_QP) and hybrid of CGA and QP (CGA_QP) for solving the constrained instances are shown in Table II. The three performance measure values for the stand-alone solvers with both C1 and C2 applied are also provided. For illustrative purposes, the AEF of each algorithm procedure is plotted along with the UEF for the five datasets in Figure 2.

From the results it can be concluded that PBIL_QP demonstrates a superiority compared to CGA_QP in all datasets in terms of solution quality, having the smallest APE value. From the graph, it also can be seen that PBIL_QP exhibits the closest approximation to, and best coverage of, the UEF. The plots reveal that PBIL_QP can provide better approximations of the constrained EFs. However we should note that this superiority comes at the cost of computational effort compared to the stand-alone solver in terms of CPU time. On the other

TABLE II
PERFORMANCE OF THE HYBRID SOLVER AGAINST PURE QP SOLVER FOR PROPORTION AND CARDINALITY CONSTRAINED PO

Datasets	N	Measures	PBIL_QP	CGA_QP	QP_ Solver
HangSeng	31	MED	0.000024	0.000269	0.0000325810
		APE	0.021058034	0.100927	0.02114048
		AE	2.17	5.35	0.031
DAX100	85	MED	0.000004	0.000286	0.0000068794
		APE	0.063872186	0.437110	0.07559099
		AE	21.75	40.9128	5.080
FTSE100	89	MED	0.000115	0.000100	0.0000904715
		APE	0.033354555	0.17317566	0.02563973
		AE	22.50	24.4492	41.294
S&P100	98	MED	0.000012	0.000391	0.0000180102
		APE	0.060978341	0.64493476	0.061856232
		AE	30.34	33.32	68.719
Nikkei225	225	MED	0.00036	0.0054	0.0003747272
		APE	0.008513889	0.8719	0.010365093
		AE	141.66	200.54	0.591

hand, CGA_QP performs rather poorly in terms of quality and convergence speed. It is even more apparent in Figure 2 that for S&P100 and Nikkei225, CGA converges too quickly and gets easily trapped in local optima. This may be due to the sample size of the population in each iteration being 2 which can reduce the diversity in searching the global solution space.

Compared to the stand-alone solver with the same limited computational time, PBIL_QP can obtain a rather satisfactory improvement over pure CPLEX in terms of APE in some of the datasets, while CPLEX obtained a slightly better result in FTSE100. Moreover, we can observe that, with the power of the latest version of the commercial solver, the gap between the solver and hybrid approach is not very large, where CPLEX can find good solutions with competitive computational effort, except for some specific risk return levels in the problem instance. This encourages us to incorporate more complex practical constraints into hybrid methods using the power of the exact methods in the solver.

We observed that the evolution progress in our hybrid approach eventually converges upon a limited number of points. For illustrative purpose, among 2000 points, we tested on the point at middle part of the EF (namely EF[1000]), for all of the 5 datasets to monitor the change in the number of distinct binary vectors generated in each generation. From Figure 3, it can be seen that after a small number of generations there are less than 10% distinct binary solutions generated in later generations. We consider the point EF[1000] for N225 as example. The portfolios converged after 171 generations. In the last generation there are 9 distinct binary solutions generated and CPLEX successfully obtains optimal real-valued solutions for each binary solution. The solutions obtained all only differ from the best one in 1 selected asset. This information may have value to the investor, offering alternative good choices in practice with only a minor modification to the portfolio.

In general, for the cardinality and quantity constrained PO problem, compared to pure solvers, the hybrid strategy still obtains improvements in terms of searching for feasible and good solutions. The CPLEX solver is normally efficient for this problem except that it requires significant computational effort to search for optimal solutions in some instances. Comparing PBIL and CGA, it can be seen that PBIL has a superior performance to CGA under the same computational environment. Since PBIL_QP succeeds in all of the scenarios, we used the results from PBIL_QP to compare against other similar work in the next section.

E. Comparison with other work

Some other work in the literature used a similar hybrid strategy of metaheuristics and exact methods, such as Moral-Escudero et.al (2006) [17], and Ruiz-Torrubiano and Suarez (2010) [22]. Both of these employed a specific genetic algorithm with different strategies to hybridise with the QP solver.

They also measured the quality of results using APE by comparing the risk obtained with a prescribed return level. All of the constraint settings are the same. We take their best results obtained for comparison in Table III. Referring to APE, we notice that the results favour our algorithm, in that the solution quality improves significantly for all of the five datasets. In terms of computational time, while the other two are comparable with the same number of points collected and similar settings, because of insufficient information and potentially different choices of points on the frontiers, an empirical comparison against our approach is difficult. The computational cost of our approach is mainly due to the number of evaluations by the solver for each candidate solution of the population in each iteration, resulting into a huge amount of time in total. In particular the solution time of the solver increases considerably with a larger problem size. Reformulating the solution encoding scheme may reduce the size of the search space. A dynamic updating scheme for PV will also be studied in our future work.

IV. Conclusion and Future Work

In this study we have demonstrated that by effectively integrating metaheuristic and exact techniques, the performance of hybrid algorithms can be improved for solving the constrained portfolio optimization problem. We have tested our proposed hybrid algorithms on the five benchmark datasets in the OR-Library, in terms of maximising total return given a prescribed risk or vice versa. The success of our approach depends on the optimality obtained in the slave solver's procedure. However, unsurprisingly, the time taken is still quite large. Needless to say, with various criteria and objectives, there is no universally recommendable algorithm. As observed, the development of commercial solvers for exact methods provides a research opportunity to test on a larger scale problem and incorporate more complicated real world constraints.

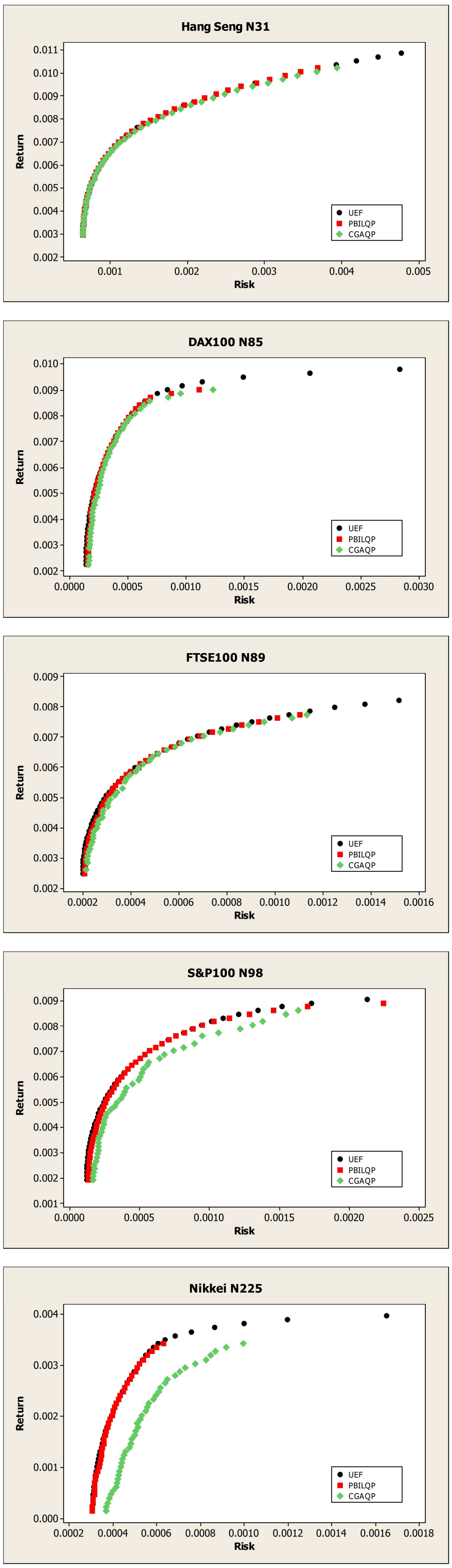

Fig. 2. AEF for constrained PO for five datasets

TABLE III
COMPARISON OF EXISTING APPROACHES BY THE AVERAGE PERCENTAGE ERROR (APE) FOR THE PROBLEM WITH $K = 10$, $\varepsilon = 0.01$, $\delta = 1$

Index	Hybrid PBILQP	Moral-Escudero et al. (2006)	Ruiz-Torrubiano and Suarez(2010)
Hang Seng	0.021058034	0.00321150	0.003211150
DAX 100	0.063872186	2.53180074	2.53162860
FTFS 100	0.033354555	1.92150019	1.92152413
S&P 100	0.060978341	4.69506892	4.69373181
Nikkei 225	0.008513889	0.20197748	0.20197748

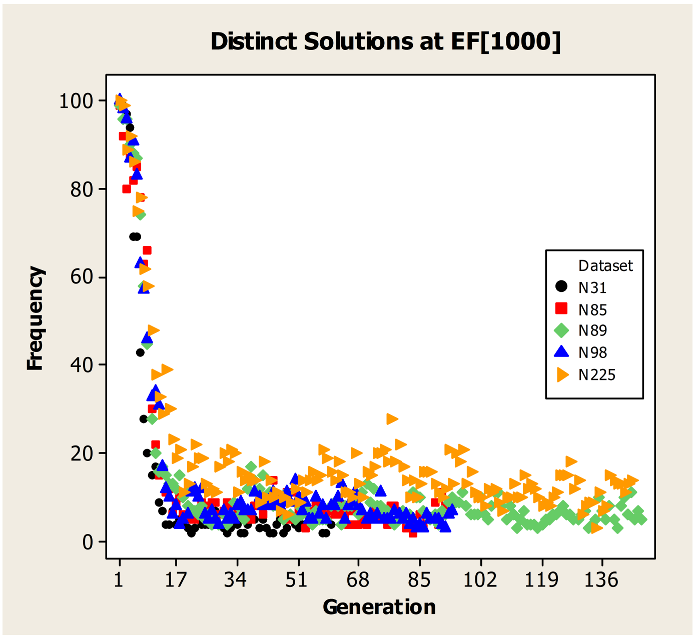

Fig. 3. Distinct binary solution through generations

In our future work, we plan to develop more efficient procedures while keeping a high level of solution quality. These include problem remodelling and adaptive strategies for searching the global optimum. In addition, the transaction cost is not currently considered. It will be interesting to include this in the model to investigate the strength of the hybrid heuristic methods when dealing with other complex real world constraints.

REFERENCES

[1] H. Markowitz, "Portfolio selection," *The Journal of Finance*, vol. 7, no. 1, pp. pp. 77–91, 1952.
[2] R. Mansini, W. Ogryczak, and M. G. Speranza, "Twenty years of linear programming based portfolio optimization," *European Journal of Operational Research*, vol. 234, no. 2, pp. 518 – 535, 2014, 60 years following Harry Markowitz's contribution to portfolio theory and operations research.
[3] G. di Tollo and A. Roli, "Metaheuristics for the portfolio selection problem," *International Journal of Operationas Research*, vol. 5, no. 1, pp. 13–35, 2008.
[4] D. Maringer, "Heuristic optimization for portfolio management [application notes]," *Computational Intelligence Magazine, IEEE*, vol. 3, no. 4, pp. 31 –34, November 2008.
[5] N. Jobst, M. Horniman, C. Lucas, and G. Mitra, "Computational aspects of alternative portfolio selection models in the presence of discrete asset choice constraints," *Quantitative Finance*, vol. 1, no. 5, pp. 489–501, 2001.
[6] H. Konno and H. Yamazaki, "Mean-absolute deviation portfolio optimization model and its applications to tokyo stock market," *Management Science*, vol. 37, no. 5, pp. 519–531, May 1991.
[7] R. Mansini and M. Speranza, "Heuristic algorithms for the portfolio selection problem with minimum transaction lots," *European Journal of Operational Research*, vol. 114, no. 2, pp. 219 – 233, 1999.
[8] D. Goldfarb and G. Iyengar, "Robust portfolio selection problems," *Mathematics of Operations Research*, vol. 28, pp. 1–38, 2001.
[9] R. Jansen and R. van Dijk, "Optimal benchmark tracking with small portfolios," *The Journal of Portfolio Management*, vol. 28, no. 2, pp. 33–39, 2002.
[10] M. Lobo, L. Vandenberghe, S. Boyd, and H. Lebret, "Applications of second-order cone programming," *Linear Algebra and its Applications*, vol. 284, no. 1-3, pp. 193 – 228, 1998.
[11] A. Fernandez and S. Gamez, "Portfolio selection using neural networks," *Computers & Operations Research*, vol. 34, no. 4, pp. 1177 – 1191, 2007.
[12] T. Cura, "Particle swarm optimization approach to portfolio optimization," *Nonlinear Analysis: Real World Applications*, vol. 10, no. 4, pp. 2396 – 2406, 2009.
[13] T.-J. Chang, N. Meade, J. Beasley, and Y. Sharaiha, "Heuristics for cardinality constrained portfolio optimisation," *Computers & Operations Research*, vol. 27, no. 13, pp. 1271 – 1302, 2000.
[14] O. Rifki and H. Ono, "A survey of computational approaches to portfolio optimization by genetic algorithms," in *18th International Conference Computing in Economics and Finance*. Society for Computational Economics, June 2012.
[15] G. Deng and W. Lin, "Ant colony optimization for markowitz mean-variance portfolio model," in *Swarm, Evolutionary, and Memetic Computing*, ser. Lecture Notes in Computer Science, B. Panigrahi, S. Das, P. Suganthan, and S. Dash, Eds. Springer Berlin Heidelberg, 2010, vol. 6466, pp. 238–245.
[16] L. Di Gaspero, G. di Tollo, A. Roli, and A. Schaerf, "Hybrid metaheuristics for constrained portfolio selection problems," *Quantitative Finance*, vol. 11, no. 10, pp. 1473–1487, 2011.
[17] R. Moral-Escudero, R. Ruiz-Torrubiano, and A. Suarez, "Selection of optimal investment portfolios with cardinality constraints," in *IEEE Congress on Evolutionary Computation*, 2006, pp. 2382 –2388.
[18] S. Baluja, "Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning," Computer Science Department, Pittsburgh, PA, Tech. Rep. CMU-CS-94-163, 1994.
[19] G. Harik, F. Lobo, and D. Goldberg, "The compact genetic algorithm," *Evolutionary Computation, IEEE Transactions on*, vol. 3, no. 4, pp. 287–297, Nov 1999.
[20] J. E. Beasley, "Or-library: Distributing test problems by electronic mail," *Journal of the Operational Research Society*, vol. 41, pp. 1069–1072, 1990.
[21] A. Chen, Y. Liang, and C. Liu, "An artificial bee colony algorithm for the cardinality-constrained portfolio optimization problems," in *2012 IEEE Congress on Evolutionary Computation (CEC)*, 10-15 June 2012, pp. 1–8.
[22] R. Ruiz-Torrubiano and A. Suarez, "Hybrid approaches and dimensionality reduction for portfolio selection with cardinality constraints," *Computational Intelligence Magazine, IEEE*, vol. 5, no. 2, pp. 92–107, May 2010.