Constraint Handling Methods for Portfolio Optimization using Particle Swarm Optimization

Stuart G. Reid
Department of Computer Science
University of Pretoria
South Africa
Email: stuartgordonreid@gmail.com

Katherine M. Malan
Department of Computer Science
University of Pretoria
South Africa
Email: kmalan@cs.up.ac.za

***Abstract*—Given a portfolio of securities, portfolio optimization aims to optimize the proportion of capital allocated to each security such that either the risk of the portfolio is minimized for a given level of expected return, expected return is maximized for a given risk budget, or the risk-adjusted expected return of the portfolio is maximized. Extensions to the portfolio optimization problem can result in it becoming more difficult to solve which has prompted the use of computational intelligence optimization methods over classical optimization methods. The portfolio optimization problem is subject to two primary constraints namely, that all of the capital available to the portfolio should be allocated between the constituent securities and that the portfolio remain long only and unleveraged. Two popular methods for finding feasible solutions when using classical optimization methods are the penalty function and augmented Lagrangian methods. This paper presents two new constraint handling methods namely, a portfolio repair method and a preserving feasibility method based on the barebones particle swarm optimization (PSO) algorithm. The purpose is to investigate which constraint handling techniques are better suited to the problem solved using PSO. It is shown that the particle repair method outperforms traditional constraint handling methods in all tested dimensions whereas the performance of the preserving feasibility method tends to deteriorate as the dimensionality of the portfolio optimization problem is increased.**

I. Introduction

Harry Markowitz famously established the basis of portfolio optimization in his seminal article, *Portfolio Selection*, in 1952 [1]. Subsequently, the portfolio optimization problem has been extended in order to make it more realistic [2] and robust [3]. Such extensions can result in the portfolio optimization problem becoming more difficult to solve [2][4] which in turn has prompted the use of computational intelligence optimization methods over classical optimization methods [5]. Various computational intelligence paradigms have been used to solve realistic and robust portfolio optimization problems [5]. In particular, swarm intelligence algorithms such as particle swarm optimization (PSO) [6][7][8][9][10] and ant colony optimization [11] have shown good results on these problems.

The specification of the portfolio optimization problem by Markowitz defined two constraints. The first constraint was a hard equality constraint which specified that all of the capital available to the portfolio and no more than that should be allocated between the constituent securities. The second constraint was a boundary constraint which specified that the portfolio was long only. Mathematically speaking this means that the proportion of capital allocated to each security must be positive. Two popular methods for finding feasible solutions when using classical optimization methods are the penalty function and augmented Lagrangian methods [12]. This paper presents two new constraint handling methods which are potentially better suited to computational intelligence optimization methods namely, a portfolio repair method and a preserving feasibility method for the barebones PSO algorithm.

Related studies in the use of PSO for portfolio optimization [6][7][8][9][10] have focused on the good performance of PSO in contrast to other algorithms, such as genetic algorithms. The contribution of this paper is to investigate different constraint handling approaches using the same PSO algorithm. Instead of implementing a single constraint handling method, the standard barebones PSO algorithm is implemented with four different constraint handling approaches. In this way, the effect of the constraint handling technique on performance is isolated from the effect of the optimization algorithm.

The remainder of this paper is structured as follows. Section II defines the portfolio optimization problem. Section III introduces the barebones particle swarm optimization algorithm. Section IV defines the three constraint handling techniques. Section V describes the experiments conducted in this study. Section VI presents the results obtained through those experiments and lastly, Section VII concludes and suggests related future research topics.

II. Portfolio Optimization

A portfolio, P, consists of capital, n securities, $S_1, \ldots, S_n$, and n weights, $w_1, \ldots, w_n$, where weight w_j represents the proportion of capital which is allocated to security S_j. For any given security, S_j, the expected return of that security, $E(R_j)$, is equal to its mean historical return and the risk of that security, σ_j, is equal to the standard deviation of its historical returns. Using modern portfolio theory [1] the expected return and risk for the portfolio are calculated as follows,

$$E(R_P) = \sum_{j=1}^{n} w_j E(R_j) \tag{1}$$

$$\sigma_P = \sqrt{\sum_{j=1}^{n} w_j^2 \sigma_j^2 + \sum_{j=1}^{n} \sum_{k=1, k \neq j}^{n} w_j w_k \sigma_j \sigma_k \rho_{jk}} \tag{2}$$

978-1-4799-7560-0/15 $31.00 © 2015 IEEE
DOI 10.1109/SSCI.2015.246

where ρ_{jk} is the correlation coefficient between the historical returns of securities S_j and S_k. Portfolio optimization has as its objective to optimize the weights of the portfolio such that either (A) the risk of the portfolio, σ_P, is minimized for a given level of expected return; (B) expected return, $E(R_P)$, is maximized for a given risk budget; or (C) the risk-adjusted expected return of the portfolio is maximized. One measure of risk-adjusted return is the Sharpe ratio [13], SR_P. The Sharpe ratio divides the units of expected excess return above the risk-free rate of return, R_f, generated by the portfolio by the units of risk required to obtain those expected excess returns,

$$SR_P = \frac{E(R_P) - R_f}{\sigma_P}. \tag{3}$$

The objective function for the portfolio optimization problem when optimizing objective (C) can be expressed as the following minimization function,

$$\min f(w) \text{ where } f(w) = -SR_P \tag{4}$$

$$\text{s.t. } \sum_{j=1}^{n} w_j = 1.0 \text{ and} \tag{5}$$

$$w_j \geq 0.0 \ \forall \ j \in \{1, \ldots, n\}. \tag{6}$$

The constraints are interesting because they cause all of the feasible solutions to exist within an $(n-1)$-dimensional simplex inside the n dimensional search space.

As such, the feasible search space is a single point when $n = 1$, a straight line when $n = 2$ (see Figure 1), a triangular hyperplane when $n = 3$ (see Figure 2) and a tetrahedron when n = 4. In general when $n \in \mathbb{Z}$ the feasible search space is an $(n-1)$-dimensional standard simplex defined by,

$$w_1, \cdots, w_{n-1} \geq 0 \tag{7}$$

$$w_1 + \cdots + w_{n-1} \leq 1 \tag{8}$$

$$w_n = 1 - w_1 - \cdots - w_{n-1}. \tag{9}$$

The above statements assume that the two constraints are strictly applied. In reality the constraints could be relaxed. If the securities were sold short, meaning that the securities were sold prior to being bought, this would result in a negative weight and if the portfolio was leveraged, meaning that funds in excess of the available capital were borrowed to invest in the securities, this would result in a capital allocation greater than 100% or 1. In these scenarios the feasible search space would extend beyond the $(n-1)$-dimensional simplex.

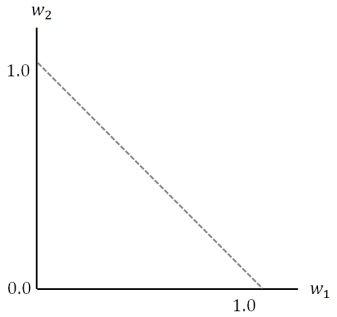

Fig. 1. When $n = 2$ the feasible space is a straight line (1-dimensional simplex) between the points $(w_1, w_2) = (1.0, 0.0)$ and $(w_1, w_2) = (0.0, 1.0)$.

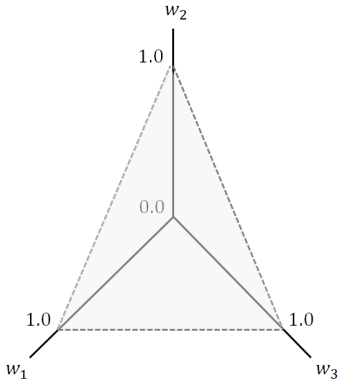

Fig. 2. When $n = 3$ the feasible space is a triangular hyperplane (2-dimensional simplex) between the points $(w_1, w_2, w_3) = (1.0, 0.0, 0.0)$, $(w_1, w_2, w_3) = (0.0, 1.0, 0.0)$, and $(w_1, w_2, w_3) = (0.0, 0.0, 1.0)$.

III. Barebones particle swarm optimization

This section describes the standard barebones particle swarm optimization algorithm for portfolio optimization.

A. Particle Swarm Optimization

The PSO algorithm is a population based search algorithm originally inspired by the social behaviour of flocking birds [14]. In a PSO, a swarm, X, of m particles is randomly initialized. Each particle in the swarm, $\mathbf{x}_i$, is encoded as an n-dimensional solution,

$$\mathbf{x}_i = (x_{ij}), \ \forall j = 1, 2, ..., n. \tag{10}$$

For the portfolio optimization problem, each vector corresponds to the weights assigned to the constituent securities in the portfolio, that is,

$$x_{ij} = w_j. \tag{11}$$

In addition to its current weight vector, each particle, i, maintains a reference to its personal best weight vector, $\mathbf{y}_i$.

For each iteration of the algorithm, the global best weight vector, $\hat{\mathbf{y}}$, is defined as the best of all the particles' personal bests. For each particle, the vector of weights is updated by adding a velocity term, $\mathbf{v}_i$, to each weight, that is

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t+1) \tag{12}$$

where, $\mathbf{x}_i(t)$ is the weight vector associated with the portfolio at time t, and $\mathbf{v}_i(t+1)$ is the velocity vector update required for $\mathbf{x}_i$ at time $t+1$, computed as,

$$\begin{aligned} v_{ij}(t+1) = w.v_{ij}(t) + c_1.r_{1j}(t).[y_{ij}(t) - x_{ij}(t)] \\ +c_2.r_{2j}(t).[\hat{y}_j(t) - x_{ij}(t)] \end{aligned} \tag{13}$$

where w is the inertia weight; c_1 and c_2 are positive acceleration coefficients to scale the contribution from the cognitive component, $r_{1j}(t).[y_{ij}(t) - x_{ij}(t)]$, and the social component, $r_{2j}(t).[\hat{y}_j(t) - x_{ij}(t)]$, respectively; $v_{ij}(t)$ is the j^{th} component of the previous velocity calculated for $\mathbf{x}_i$; and lastly $r_{1j}(t)$ and $r_{2j}(t)$ are uniformly distributed random numbers, $r_{1j}(t), \ r_{2j}(t) \sim U(0, 1)$.

Formal proofs [15][16][17] have shown that each particle converges to a point that is a weighted average between the

personal best and neighbourhood best positions. If $c_1 = c_2$ then each dimension of each particle converges to the midpoint, **mp**, between the particle's personal best position and the global best position,

$$mp_j(t) = \frac{y_{ij}(t) + \hat{y}_j(t)}{2}. \quad (14)$$

This supports a simplification of the PSO proposed by Kennedy called the barebones PSO [18] in which new particles' positions are sampled from a Gaussian distribution with mean equal to $mp_j(t)$ and with standard deviation equal to

$$\sigma = \left|y_{ij}(t) - \hat{y}_j(t)\right|. \quad (15)$$

In the barebones PSO, the position at time $t+1$ is then sampled from the following Gaussian distribution,

$$x_{ij}(t+1) \sim \mathcal{N}\left(\frac{y_{ij}(t) + \hat{y}_j(t)}{2}, \left|y_{ij}(t) - \hat{y}_j(t)\right|\right). \quad (16)$$

B. Barebones PSO

Algorithm 1 provides pseudo-code for the barebones PSO which does not use any constraint handling methods. Note that the Dirichlet distribution referred to in the initialization step is described later in Section IV-D.

Data: Portfolio of simulated assets, $P(0)$
Result: Optimized portfolio, $P(T)$
Initialize swarm, X, consisting of m feasible particles sampled from a flat Dirichlet distribution;
Initialize the personal best positions, Y, equal to the initial positions for each particle, $Y = X$;
for $t \leftarrow 0$ **to** T **do**
  $\hat{\mathbf{y}} \leftarrow \mathbf{y}_1$
  for $i \leftarrow 1$ **to** m **do**
    if $f(\boldsymbol{y}_i) < f(\hat{\mathbf{y}})$ **then**
      $\hat{\mathbf{y}} \leftarrow \mathbf{y}_i$ // Get global best position
    end
  end
  for $i \leftarrow 1$ **to** m **do**
    update $\mathbf{x}_i$ using equation (16);
    if $f(\boldsymbol{x}_i) < f(\boldsymbol{y}_i)$ **then**
      $\mathbf{y}_i \leftarrow \mathbf{x}_i$ // Update personal best positions
    end
  end
end

Algorithm 1: Standard barebones particle swarm optimization using no constraint handling method

The problem with this algorithm is that the PSO, more often than not, converges on an infeasible solution. As such, constraint handling techniques are used in order to guarantee that the PSO returns a feasible solution to the problem.

IV. Constraint Handling

This section defines the four constraint handling methods considered in this paper namely the, particle repair, penalty function, augmented Lagrangian, and preserving feasibility methods. Each method works by augmenting the portfolio optimization problem in such a way that the final solution is either guaranteed or more likely to satisfy the constraints. The particle repair method augments the particle weight vectors, the penalty and augmented Lagrangian methods augment the objective function, and the preserving feasibility method augments the operators in the barebones PSO algorithm.

A. Particle Repair Method

The particle repair method works by normalizing the maximum of each weight and a very small term, ϵ, in the solution by the sum of the maximums of each weight and ϵ in the solution for each particle,

$$x_{ik} = \frac{\max\left(x_{ik}, \epsilon\right)}{\sum_{j=1}^{n} \max\left(x_{ij}, \epsilon\right)} \;\forall\; k \in (1, 2, ..., n). \quad (17)$$

The use of the maximums guarantees that the constraint handling method does not try to divide by zero in the unlikely that event that each weight in the solution equals 0. Algorithm 2 provides pseudo-code for a barebones PSO which uses the particle repair constraint handling technique.

Data: Portfolio of simulated assets, $P(0)$
Result: Optimized portfolio, $P(T)$
Initialize swarm, X, consisting of m feasible particles sampled from a flat Dirichlet distribution;
Initialize the personal best positions, Y, equal to the initial positions for each particle, $Y = X$;
for $t \leftarrow 0$ **to** T **do**
  $\hat{\mathbf{y}} \leftarrow \mathbf{y}_1$
  for $i \leftarrow 1$ **to** m **do**
    if $f(\boldsymbol{y}_i) < f(\hat{\mathbf{y}})$ **then**
      $\hat{\mathbf{y}} \leftarrow \mathbf{y}_i$ // Get global best position
    end
  end
  for $i \leftarrow 1$ **to** m **do**
    update $\mathbf{x}_i$ using equation (16);
    repair $\mathbf{x}_i$ using equation (17);
    if $f(\boldsymbol{x}_i) < f(\boldsymbol{y}_i)$ **then**
      $\mathbf{y}_i \leftarrow \mathbf{x}_i$ // Update personal best positions
    end
  end
end

Algorithm 2: Barebones particle swarm optimization using the particle repair constraint handling method

B. Penalty Function Method

The penalty function method works by converting the constrained portfolio optimization problem into an unconstrained optimization problem by adding terms to the objective function. These terms represent the degree to which the solution violates the constraints [4]. For the portfolio optimization problem two penalty functions are added to the objective function. The first term relates to the equality constraint, $C_E(\mathbf{x}_i)$, and the second term relates to the boundary constraint, $C_B(\mathbf{x}_i)$. These functions are given by,

$$C_E(\mathbf{x}_i) = 1.0 - \sum_{j=1}^{n} x_{ij} \quad (18)$$

$$C_B(\mathbf{x}_i) = \sum_{j=1}^{n} b(x_{ij}) \tag{19}$$

$$\text{where } b(x_{ij}) = \begin{cases} |x_{ij}|, & \text{if } x_{ij} < 0.0 \\ 0, & \text{otherwise.} \end{cases} \tag{20}$$

The updated objective function becomes,

$$\min \Phi(\mathbf{x}_i, t) = f(\mathbf{x}_i) + \mu_E(t) C_E(\mathbf{x}_i)^2 + \mu_B(t) C_B(\mathbf{x}_i)^2 \tag{21}$$

where $\mu_E(t)$ and $\mu_B(t)$ are time-dependent constraint penalty coefficients for functions $C_E(\mathbf{x}_i)$ and $C_B(\mathbf{x}_i)$. These parameters are typically initialized to 1.0 and then compounded at a rate of g per iteration, in other words,

$$\mu_E(t+1) = g \times \mu_E(t) \text{ and} \tag{22}$$

$$\mu_B(t+1) = g \times \mu_B(t). \tag{23}$$

The penalty terms are squared so that the resulting fitness landscape is smooth. Whilst smoothing matters for gradient-based search algorithms it may matter less for heuristic-based search algorithms such as the Barebones PSO.

Algorithm 3 provides pseudo-code for a barebones PSO which uses the penalty function constraint handling method.

Data: Portfolio of simulated assets, $P(0)$
Data: Penalty coefficients starting values, $\mu_E(0)$ and $\mu_B(0)$
Result: Optimized portfolio, $P(T)$
Initialize swarm, X, consisting of m feasible particles sampled from a flat Dirichlet distribution;
Initialize the personal best positions, Y, equal to the initial positions for each particle, $Y = X$;
for $t \leftarrow 0$ **to** T **do**
  $\hat{\mathbf{y}} \leftarrow \mathbf{y}_1$
  for $i \leftarrow 1$ **to** m **do**
    if $\Phi(\boldsymbol{y}_i, t) < \Phi(\hat{\mathbf{y}}, t)$ **then**
      $\hat{\mathbf{y}} \leftarrow \mathbf{y}_i$ // Get global best position
    end
  end
  for $i \leftarrow 1$ **to** m **do**
    update $\mathbf{x}_i$ using equation (16);
    if $\Phi(\boldsymbol{x}_i, t) < \Phi(\boldsymbol{y}_i, t)$ **then**
      $\mathbf{y}_i \leftarrow \mathbf{x}_i$ // Update personal best positions
    end
  end
  update $\mu_E(t)$ using equation (22);
  update $\mu_B(t)$ using equation (23);
end

Algorithm 3: Barebones particle swarm optimization using the penalty function constraint handling method

C. Augmented Lagrangian Method

The augmented Lagrangian method is similar to the penalty function method except that only half the squared penalty term minus the penalty term is added to the objective function for each constraint [4]. The updated objective function becomes,

$$\begin{aligned} \min \Phi(\mathbf{x}_i, t)_L &= f(\mathbf{x}_i) \\ &+ \tfrac{1}{2}\mu_E(t) C_E(\mathbf{x}_i)^2 - \lambda_E(t) C_E(\mathbf{x}_i) \\ &+ \tfrac{1}{2}\mu_B(t) C_B(\mathbf{x}_i)^2 - \lambda_B(t) C_B(\mathbf{x}_i) \end{aligned} \tag{24}$$

where $\lambda_E(t)$ and $\lambda_B(t)$ are time-dependent coefficients which estimate the Lagrangian multiplier of the objective function. These terms are updated each iteration as follows,

$$\lambda_E(t+1) = \lambda_E(t) - \mu_E(t) C_E(\mathbf{x}_i) \tag{25}$$

$$\lambda_B(t+1) = \lambda_B(t) - \mu_B(t) C_B(\mathbf{x}_i). \tag{26}$$

Algorithm 4 provides pseudo-code for a barebones PSO which uses the augmented Lagrangian constraint handling methods.

Data: Portfolio of simulated assets, $P(0)$
Data: Penalty coefficients starting values, $\mu_E(0)$ and $\mu_B(0)$
Data: Lagrangian coefficients starting values, $\lambda_E(0)$ and $\lambda_E(0)$
Result: Optimized Portfolio, $P(T)$
Initialize swarm, X, consisting of m feasible particles sampled from a flat Dirichlet distribution;
Initialize the personal best positions, Y, equal to the initial positions for each particle, $Y = X$;
for $t \leftarrow 0$ **to** T **do**
  $\hat{\mathbf{y}} \leftarrow \mathbf{y}_1$
  for $i \leftarrow 1$ **to** m **do**
    if $\Phi(\boldsymbol{y}_i, t)_L < \Phi(\hat{\mathbf{y}}, t)_L$ **then**
      $\hat{\mathbf{y}} \leftarrow \mathbf{y}_i$ // Get global best position
    end
  end
  for $i \leftarrow 1$ **to** m **do**
    update $\mathbf{x}_i$ using equation (16);
    if $\Phi(\boldsymbol{x}_i, t)_L < \Phi(\boldsymbol{y}_i, t)_L$ **then**
      $\mathbf{y}_i \leftarrow \mathbf{x}_i$ // Update personal best positions
    end
  end
  update $\mu_E(t)$ using equation (22);
  update $\mu_B(t)$ using equation (23);
  update $\lambda_E(t)$ using equation (25);
  update $\lambda_B(t)$ using equation (26);
end

Algorithm 4: Barebones particle swarm optimization using the augmented Lagrangian constraint handling method

D. Preserving Feasibility Method

One implication of converting the constrained portfolio optimization problem into an unconstrained optimization problem using either the penalty function method or the augmented Lagrangian method is that the optimization algorithm must search over an n dimensional fitness landscape rather than the smaller $(n-1)$-dimensional simplex.

The particle repair method overcomes this by translating infeasible solutions into feasible solutions whereas the preserving feasibility method augments the barebones PSO's update operator such that the particles, once initialized within the feasible region, never leave that region. This subsection introduces a useful initialization strategy and then describes how the feasibility of the swarm can be preserved by augmenting the barebones PSO.

A convenient initialization strategy for a portfolio consisting of two or more assets is to sample the starting weight

vector for each particle from a Dirichlet distribution [19] of order $n \geq 2$. A Dirichlet distribution can be defined in terms of the Gamma function, Γ, using the parameters $\alpha_1, \ldots, \alpha_n > 0$ and n as follows,

$$Dir\left(\mathbf{x}_i; \alpha_1, \cdots, \alpha_n\right) = \frac{1}{\mathrm{B}(\alpha)} \prod_{j=1}^{n} x_{ij}^{\alpha_j - 1} \tag{27}$$

where $\mathrm{B}()$ is the multinomial Beta function, which can be expressed in terms of the gamma function as follows,

$$\mathrm{B}(\boldsymbol{\alpha}) = \frac{\prod_{j=1}^{n} \Gamma(\alpha_j)}{\Gamma\left(\sum_{j=1}^{n} \alpha_j\right)}. \tag{28}$$

Any Dirichlet distribution is convenient for portfolio optimization because it has the following support,

$$x_{ij} \in (0, 1) \text{ and } \sum_{j=1}^{n} x_{ij} = 1 \tag{29}$$

meaning that every weight vector which could be sampled from the distribution with a non-zero probability, satisfies the constraints specified in the portfolio optimization problem.

Furthermore, when $a_i = a_j \ \forall \ i, j \in n, n$ the Dirichlet distribution is said to be symmetric and the probability density function for a random weight vector, $\mathbf{x}_i$, simplifies to,

$$Dir(\mathbf{x}_i; \alpha) = \frac{\Gamma(\alpha n)}{\Gamma(\alpha)^n} \prod_{j=1}^{n} x_{ij}^{\alpha - 1} \tag{30}$$

where α is referred to as the concentration parameter. $\alpha = 1.0$ is a special case of the Dirichlet distribution which causes it to be uniform over the $(n-1)$-dimensional simplex defined by (7), (8), and (9). This is beneficial because all feasible weight vectors have a uniform probability of being sampled,

$$Dir(x_{ij}; \alpha) = Dir(x_{ik}; \alpha) \ \forall \ j, k \in n, n \tag{31}$$

meaning that when used for swarm initialization, the flat Dirichlet distribution guarantees that the swarm will be unbiased on average at iteration $t = 0$.

Another useful characteristic of the Dirichlet distribution for portfolio optimization is that when α is set equal to a particle position, $\alpha = \mathbf{x}_i$, random vectors sampled from the resulting Dirichlet distribution have a mean equal to $\mathbf{x}_i$ and a standard deviation proportional to $\mathbf{x}_i$ meaning that the Dirichlet distribution can be used to sample positions around other positions which satisfy both the equality and boundary constraints of the portfolio optimization problem.

Therefore by simply replacing the Gaussian distribution in the barebones PSO update equation (16) with the Dirichlet distribution where $\alpha = \mathbf{mp}(t)$ as calculated using equation (14), we arrive at a new feasibility preserving update equation for the barebones particle swarm optimization algorithm,

$$x_{ij}(t+1) \sim Dir\big(\mathbf{mp}(t), n\big). \tag{32}$$

Algorithm 5 provides pseudo-code for a barebones PSO which uses this preserving feasibility constraint handling method.

Data: Portfolio of simulated assets, $P(t)$
Result: Optimized portfolio, $P(t+1)$
Initialize swarm, X, consisting of m feasible particles sampled from a flat Dirichlet distribution;
Initialize the personal best positions, Y, equal to the initial positions for each particle, $Y = X$;
for $t \leftarrow 0$ **to** T **do**
  $\hat{\mathbf{y}} \leftarrow \mathbf{y}_1$
  for $i \leftarrow 1$ **to** m **do**
    if $f(\boldsymbol{y}_i) < f(\hat{\mathbf{y}})$ **then**
      $\hat{\mathbf{y}} \leftarrow \mathbf{y}_i$ // Get global best position
    end
  end
  for $i \leftarrow 1$ **to** m **do**
    update $\mathbf{x}_i$ using equation (32);
    if $f(\boldsymbol{x}_i) < f(\boldsymbol{y}_i)$ **then**
      $\mathbf{y}_i \leftarrow \mathbf{x}_i$ // Update personal best positions
    end
  end
end

Algorithm 5: Barebones particle swarm optimization using the preserving feasibility constraint handling method

V. Experiment Design

This section details the experiments and parameter configurations used to obtain the results presented in section VI.

A. Experiments Undertaken

Results were obtained by averaging the performance of each algorithm on a portfolio of n simulated assets over 60 independent optimization runs. Results were obtained for dimensions $n = \{4, 8, 16\}$. The specific parameter configurations for the algorithms are tabulated below.

Algorithm Parameter	Value
Swarm size (m)	30
Iterations	80
Independent runs	60
Minimum vector sum (ϵ)	1×10^{-8}
Constraint coefficient growth rate (g)	1.1
Equality constraint coefficient ($\mu_E(0)$)	2.0
Boundary constraint coefficient ($\mu_B(0)$)	2.0
Equality constraint multiplier ($\lambda_E(0)$)	0.5
Boundary constraint multiplier ($\lambda_B(0)$)	0.5

These parameters were selected through trial-and-error and one of the suggested areas of further research is to conduct a more exhaustive study over a wider set of parameter configurations.

B. Simulated Returns

The experiments were conducted using return sequences simulated using a Geometric Brownian Motion stochastic process (model). The Geometric Brownian Motion stochastic process assumes that security prices evolve randomly according to the following stochastic differential equation,

$$S_i(t) = \mu S_i(t) dt + \sigma S_i(t) dW(t) \tag{33}$$

where $S_i(t)$ is the price of security S_i at time t, μ represents the annualized drift of the security's price over time, dt represents time (in this paper we simulated daily returns), and $\sigma W(t)$ represents a normally distributed Wiener process with volatility σ. Solving this stochastic differential equation under

Ito's interpretation [20] we arrive at the following equation which specifies the log returns of each each asset,

$$r_i(t) = \ln\frac{S_i(t)}{S_i(0)} = \left(\mu - \frac{\sigma^2}{2}\right)t + \sigma W(t). \quad (34)$$

Because the Wiener process, $W(t)$, is normally distributed we can derive the probability density function from which log return sequences for each asset can be sampled,

$$r_i(t) \sim \mathcal{N}\left(\left(\mu - \frac{\sigma^2}{2}\right)t, \sigma^2 t\right). \quad (35)$$

The relationship between a log return, $r_i(t)$, and an arithmetic return, $R_i(t)$, is given by the following equation,

$$R_i(t) = e^{r_i(t)} - 1. \quad (36)$$

Using this model it is possible to sample n return sequences of length τ for each asset. From these return sequences the the risk, σ_i, and expected return, $E(r_i)$, of each security were calculated as was the correlation matrix. Given this information and a weight vector the portfolio risk, σ_P, expected return, $E(r_P)$, and Sharpe ratio, SR_P, were calculated as per equation (1), (2), and (3). The model was calibrated as follows,

Model parameter	Value
Annualized volatility (σ)	0.125
Annualized drift (μ)	0.08
Instantaneous time (dt)	$\frac{1}{252} \approx 0.00397$
Total time in days (τ)	500

Figure 3 shows an example of 30 securities price paths simulated using this model.

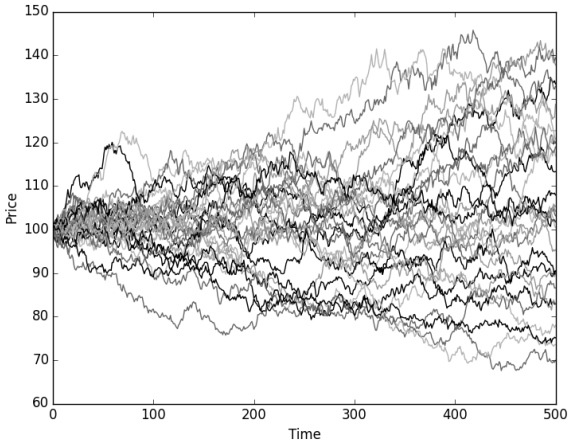

Fig. 3. Example security price paths simulated using Geometric Brownian Motion for 30 independent securities.

VI. Results

This section presents the results obtained in running the experiments discussed in Section V. The results are plotted in Figures 4 to 11. Although the algorithms were run for 250 iterations, only the first 80 iterations are plotted due to insignificant changes beyond 80 iterations. In the graphs, algorithm A1 represents the standard barebones PSO without any constraint handling.

A. Results relating to constraint violations

1) Barebones PSO: Whilst the canonical barebones PSO is able to find solutions with superior fitness values (see Figure 4) it does so by violating both the boundary and equality constraints (see Figures 5 and 6). These results support the argument that constraint handling methods are necessary when using PSO for constrained portfolio optimization problems.

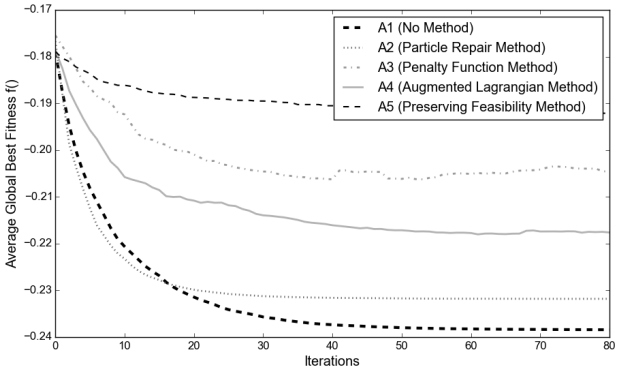

Fig. 4. Global best fitness for $n = 16$ averaged over 60 independent runs for algorithms 1 to 5 for 80 out of 250 iterations.

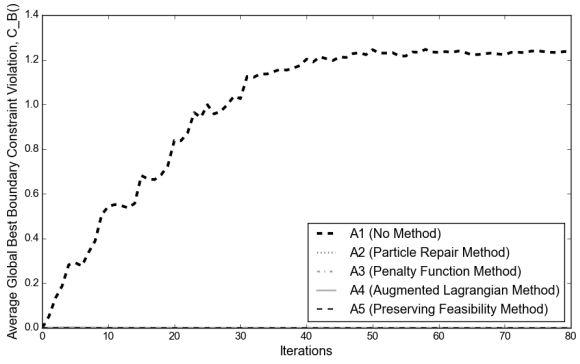

Fig. 5. Global best boundary constraint violation for $n = 16$ averaged over 60 independent runs for algorithms 1 to 5 for 80 out of 250 iterations.

2) Penalty function constraint handling method: When using the penalty function constraint handling method the PSO initially violates both the boundary and equality constraints but then starts to converge on the feasible region which causes the violations to decrease over time (see Figures 7 and 8).

3) Augmented Lagrangian constraint handling method: As with the penalty function method, when using the augmented Lagrangian constraint handling method the PSO initially violates both the boundary and equality constraints before converging on the feasible region. Relative to the penalty function method the augmented Lagrangian method violates the constraints more severely initially and then converges more slowly towards the feasible region (see Figures 7 and 8).

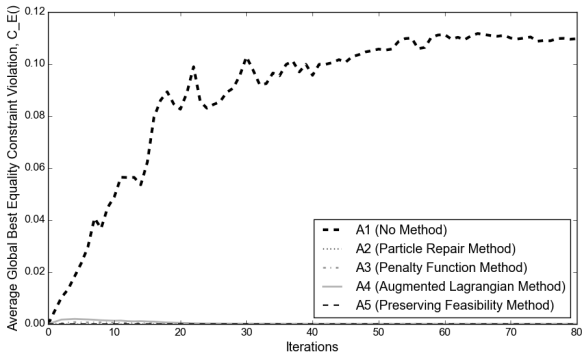

Fig. 6. Global best equality constraint violation for $n = 16$ averaged over 60 independent runs for algorithms 1 to 5 for 80 out of 250 iterations.

4) Particle repair and preserving feasibility constraint handling methods: Neither the particle repair nor the preserving feasibility constraint handling methods violate either the boundary or equality constraints (see Figures 7 and 8). This indicates that when using these constraint handling methods the particles in the swarm remain inside the feasible region.

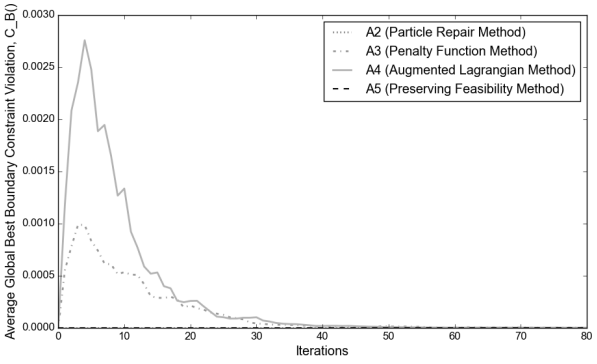

Fig. 7. Global best boundary constraint violation for $n = 16$ averaged over 60 independent runs for algorithms 2 to 5 for 80 out of 250 iterations.

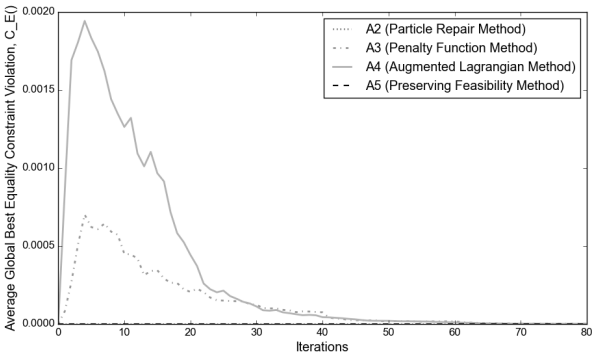

Fig. 8. Global best equality constraint violation for $n = 16$ averaged over 60 independent runs for algorithms 2 to 5 for 80 out of 250 iterations.

B. Results relating to performance

Figures 9 10 and 11 show the average global best fitness for dimensions 4, 8 and 16, respectively. From these graphs it can be seen that the particle repair constraint handling method performed the best across all three dimensions followed by the augmented Lagrangian and penalty function methods. The figures also demonstrate that the performance of the preserving feasibility constraint handling method deteriorated relative to the other constraint handling methods as the dimensionality of the problem increased. This may be caused by the Dirichlet distribution being too narrow meaning that new positions sampled from the distribution cannot reach more optimal areas of the fitness landscape (see Figure 12).

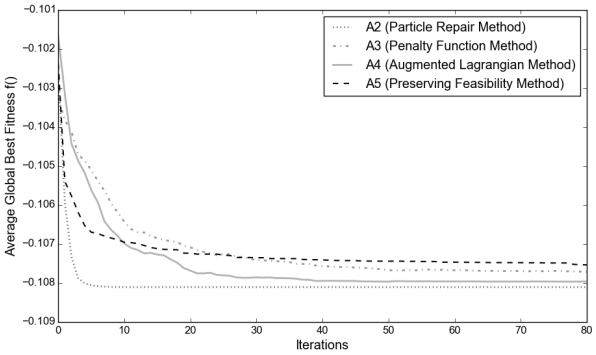

Fig. 9. Global best fitness for $n = 4$ averaged over 60 independent runs for algorithms 2 to 5 for 80 out of 250 iterations.

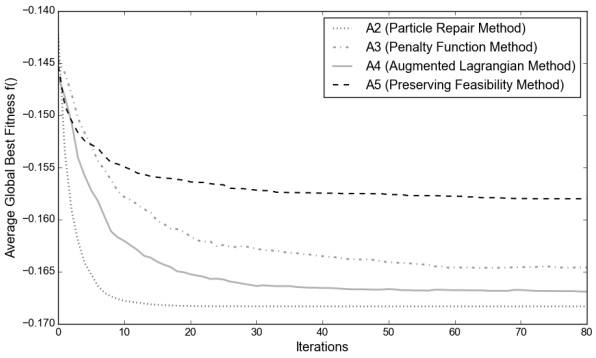

Fig. 10. Global best fitness for $n = 8$ averaged over 60 independent runs for algorithms 2 to 5 for 80 out of 250 iterations.

VII. Conclusion and Further Work

The feasible region of the fitness landscape for long-only and non-leveraged constrained portfolio optimization problems is an $(n-1)$-dimensional simplex inside the n-dimensional search space. Traditional constraint handling methods such as the penalty function method and the augmented Lagrangian method convert this constrained $(n-1)$-dimensional problem into an unconstrained n-dimensional problem by augmenting the objective function. The drawback to this approach is that

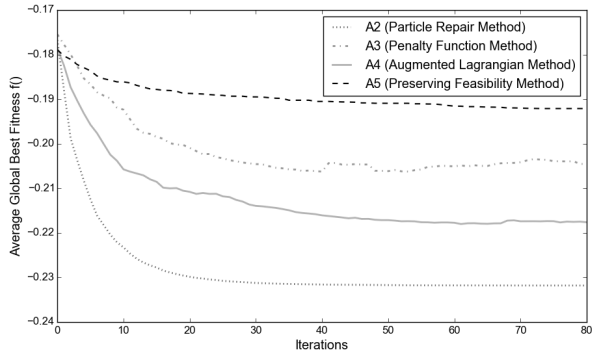

Fig. 11. Global best fitness for $n = 16$ averaged over 60 independent runs for algorithms 2 to 5 for 80 out of 250 iterations.

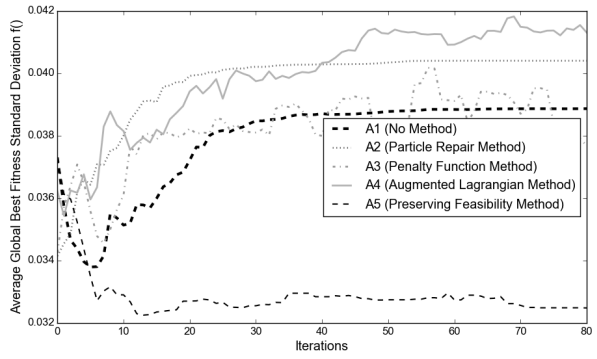

Fig. 12. Global best fitness standard deviation for $n = 16$ averaged over 60 independent runs for algorithms 1 to 5 for 80 out of 250 iterations.

the optimization algorithm much search a larger dimensional fitness landscape and is not guaranteed to produce a feasible solution. To overcome this shortcoming we proposed the use of either a new particle repair method or preserving feasibility method which augment the particles and algorithmic operators of the PSO rather than the objective function.

The results show that particle repair method performed better than traditional constraint handling methods in all three dimensions tested whereas the preserving feasibility method deteriorated relative to the other methods as the dimensionality of the portfolio optimization problem was increased. It was also shown that the augmented Lagrangian technique is superior to the penalty function method despite the fact that the use of the augmented Lagrangian technique results in relatively higher initial constraint violation and slower convergence to the feasible region. Lastly, it was shown that the PSO algorithm when used without constraint handling methods violates both the equality and boundary constraints meaning that constraint handling is an important aspect of the constrained portfolio optimization problem.

In conclusion, the particle swarm optimization algorithm is only able find quality solutions to the constrained portfolio optimization problem when constraint handling methods are applied. Avenues for further research include a more comprehensive study of the augmented Lagrangian and penalty function methods under different parameter configurations as it is expected that algorithmic tuning may improve the performance of these methods relative the particle repair method. Additionally, a study into the reachability of the barebones PSO when using the preserving feasibility method should be undertaken.

REFERENCES

[1] H. Markowitz, "Portfolio Selection", in *Journal of Finance*, vol. 7, pp. 77-91, 1952.

[2] A. Rudd and B. Rosenberg, "Realistic portfolio optimization", in *Portfolio theory 25*, pp. 21-46, 1979.

[3] F. J. Fabozzi, P. N. Kolm, and D. A. Pachamanova, "Robust portfolio optimization", in *The Journal of Portfolio Management 33.3*, pp. 40-48, 2007.

[4] M. R. Hestenes, "Multiplier and gradient methods", in *Journal of optimization theory and applications 4.5*, pp. 303-320, 1969.

[5] F. Busetti, "Metaheuristic Approaches to Realistic Portfolio Optimisation", Masters thesis, Operations Research, University of South Africa, 2000.

[6] T. Cura, "Particle swarm optimization approach to portfolio optimization", in *Nonlinear Analysis: Real World Applications*, vol. 10, pp. 2396-2406, 2009.

[7] H. Zhu, Y. Wang, K. Wang and Y. Chen, "Particle swarm optimization (PSO) for the constrained portfolio optimization problem", in *Expert Systems with Applications*, vol. 38, no. 8, pp. 10161-10169, Aug. 2011.

[8] F. Xu, W. Chen and L. Yang, "Improved Particle Swarm Optimization for Realistic Portfolio Selection", in *Proceedings of Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, pp. 185-190, 2007.

[9] G. Kendall and Y. Su, "A particle swarm optimization approach in the construction of optimal risky portfolios", in *Proceedings of the 23rd IASTED International Multi-Conference Artificial Intelligence and Applications*, February 14-16, pp. 140-145, 2005.

[10] H. R. Golmakani and M. Fazel, "Constrained portfolio selection using particle swarm optimization", in *Expert Systems with Applications*, vol. 38, no. 7, pp. 8327-8335, 2011.

[11] K. Doerner, W. J. Gutjahr, R. F. Hartl, C. Strauss and C. Stummer, "Pareto Ant Colony Optimization: A metaheuristic approach to multiobjective portfolio selection", in *Annals of operations research*, vol. 131, pp. 79-99, Oct. 2004.

[12] P. M. Pardalos, M. Sandström, and C. Zopounidis, "On the use of optimization models for portfolio selection: A review and some computational results", in *Computational Economics 7.4*, pp. 227-244, 1994.

[13] W. F. Sharpe, "The sharpe ratio", in *The journal of portfolio management 21.1*, pp. 49-58, 1994.

[14] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory", in *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pp. 39-43, 1995.

[15] I. C. Trelea, "The particle swarm optimization algorithm: convergence analysis and parameter selection", in *Information processing letters 85.6* pp. 317-325, 2002.

[16] F. Van Den Bergh, "An analysis of particle swarm optimizers", PhD Thesis, Department of Computer Science, University of Pretoria, South Africa, 2006.

[17] F. Van Den Bergh and A. P. Engelbrecht "A study of particle swarm optimization particle trajectories.", in *Information sciences 176.8* pp. 937-971, 2006.

[18] J. Kennedy, "Bare bones particle swarms", in *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, pp. 80-87, 2003.

[19] N. L. Johnson, S. Kotz, and N. Balakrishnan, "Continuous Multivariate Distributions, volume 1, Models and Applications", New York: John Wiley & Sons, 2002.

[20] A. N Borodin and P. Salminen, "Handbook of Brownian motion – facts and formulae", Birkhäuser, 2012.