Classification-assisted Memetic Algorithms for Equality-constrained Optimization Problems with Restricted Constraint Function Mapping

Stephanus Daniel Handoko, Kwoh Chee Keong, and Ong Yew Soon
School of Computer Engineering
Nanyang Technological University
Singapore
sdhandoko@ntu.edu.sg

***Abstract*—Solving constrained optimization problems using Memetic Algorithms (MAs) has become an important alternative that overcomes the following limitations: the adversary of a global search method to locate the global optimum with sufficient precision and the inability of a local search method to escape from some local optimum. With the success of MAs, researchers in the field are now focused more than ever on the aspect of efficiency of the algorithms such that it would be possible to effectively employ MAs in the context of computationally expensive optimization problems where single evaluation of the objective and constraint functions may require minutes to hours of CPU time or even more. One of the important design issues of MAs is the choice of individuals produced by the global search method upon which the local search procedure should be applied. Selecting only some potential individuals lessens the demand for functional evaluations hence accelerates convergence to the global optimum. In recent years, advances have been made targeting optimization problems with either inequality constraints $\mathbf{g}(\mathbf{x}) \leq \mathbf{0}$ or single equality constraint $h(\mathbf{x}) = 0$. The earlier utilized the feasibility class (feasible or infeasible) of previously evaluated candidate solutions while the latter made use of their raw constraint values. In the latter case, the feasible candidate solutions lie on the constraint boundary defined by the equality $h(\mathbf{x}) = 0$. The presence of previously evaluated candidate solutions with different signs of constraint values within some localities thus allows the estimation of the constraint boundary. An individual will undergo local search only if it is sufficiently close to the approximated boundary. Elegant as it may seem, the approach had unfortunately assumed that every constraint function maps the design variables to optimize into unbounded real values. This, however, may not always be the case in practice. In this paper, we present a strategy to efficiently solve equality-constrained problems; the constraint function of which maps the design variables into restricted (either strictly non-negative or strictly non-positive) real values only.**

Keywords-Evolutionary Computation; Memetic Algorithms; Genetic Algorithms; Sequential Quadratic Programing; Equality-constrained Optimization; Computationally-expensive Problems; Classification; Support Vector Machine

I. INTRODUCTION

The construction and simulation of models of the reality—whether it is in the field of economics, physics, or mathematics, or be it in the field of biology and chemistry—oftentimes involves optimization. The optimization does indeed play an indispensable role in both science and engineering with the constrained optimization being commonly encountered in practice as things in nature are generally restricted by either some common sense, such as the nonexistence of negative lengths, or the available resources, such as the maximum length to the steel bars that can be manufactured in the world today. In general, a constrained optimization problem can be written in the form of

$$\min_{\mathbf{x}} f(\mathbf{x}) \tag{1}$$

subject to

$$\mathbf{g}(\mathbf{x}) \leq \mathbf{0} \tag{2}$$

$$\mathbf{h}(\mathbf{x}) = \mathbf{0} \tag{3}$$

Whenever the objective $f(\mathbf{x})$ and the inequality and equality constraints $\mathbf{g}(\mathbf{x})$ and $\mathbf{h}(\mathbf{x})$, respectively, involve only either linear or quadratic functions, the above constrained optimization problem can be solved exquisitely as there exist systematic deterministic approaches targeted specifically at solving the linear programs, the quadratic programs, or the quadratically-constrained quadratic programs [1]. Whenever highly nonlinear functions are involved, nonetheless, there exist no such systematic deterministic approaches that would guarantee convergence to the true global optimum since highly nonlinear functions translate easily into multiple basins of attraction, hence numerous local optima. This means either exhaustive or random sampling of the search space of the

Supported by Singapore MoE AcRF Grant No. MOE2008-T2-1-1074.

problem would ultimately be required. In this context, the Memetic Algorithm (MA), which combines the stochastic Genetic Algorithms (GA) with some deterministic mathematical programming methods, is a more efficient alternative to the exhaustive search, which could potentially solve the problem but would undoubtedly be both time-consuming and resource-intensive. More details of these algorithms will be discussed in the next section.

Models are constructed and simulated to mimic behaviors of the nature as dealing with nature directly (experimentations) may simply be impossible or may sometimes be too expensive or too hazardous to perform. As computer simulations pose no risk and are generally low-cost, model simulations before real experimentations are consequently desirable. It is intuitive that complexity of the models determines their ability to mimic reality. It may be possible to perfectly mimic simple behaviors using simple models. More complex phenomenon, however, requires more complex models, as well, such that it is closer to reality when it comes to mimicking the behaviors. Complex models, unfortunately, are quite-so-often rather expensive to compute. Single evaluation may require minutes to hours of computation time or even more. Simple MA, in which global and local searches are simply interleaved one after another, may not be the most efficient strategy after all in the context of the computationally expensive optimization problems. One key design issue in MA is the selection of candidate solution for local refinement. The work presented in [2], for example, introduced a simulated heating technique for systematic integration of local search methods into the framework of evolutionary computation. In [3], a new paradigm of incorporating classification into the framework of optimization was first proposed. The classification technique is used to distinguish feasible space from the infeasible one in the context of inequality-constrained optimization [4] or positive constraint space from the negative one for single equality constraint function [5]. The latter, however, assumes that the constraint function maps a set of design variables into an unbounded real value. For optimization problems whose constraint function exhibits restricted mapping into strictly non-negative or strictly non-positive real values, an alternative strategy is unmistakably necessary. For this purpose, a novel assessment strategy based on the value of the partial derivatives of the constraint is proposed in this paper.

In what follows, this paper is organized such that literature review will be presented next in Section II, followed by the explicit problem statement in Section III. The proposed approach will then be elaborated in great details in Section IV, followed by Section V which presents results and discussions from an empirical study on the modified version of a standard benchmark problem. Section VI finally concludes the paper and proposes future directions.

II. Literature Review

As early as the 1960s, numerical techniques have been developed to solve constrained optimization problems using computers. They can be grouped into three general categories, each of which is discussed separately in the following subsections: deterministic algorithms, stochastic algorithms, and memetic algorithms.

A. Deterministic Algorithms

With rigorous mathematical background, a class of deterministic algorithms called the methods of feasible directions proceed from one feasible solution to another to exquisitely solve nonlinear programs for the Karush-Kuhn-Tucker (KKT) points [1]. Belonging to this class of algorithms are the Zoutendijk algorithm [6] as well as the sequential linear programming (SLP) [7][8] and the sequential quadratic programming (SQP) [9] approaches. While the SLP employs first-order approximations to both the objective and constraint functions and is therefore prone to slow convergence, the SQP takes advantage of the second-order approximation to the objective function to enjoy quadratic rate of convergence.

At the *i*-th major iteration of the SQP, the following quadratic program is solved for direction **d**.

$$f\left(\mathbf{x}^{(i)}\right)+\nabla f\left(\mathbf{x}^{(i)}\right)^T\mathbf{d}+\frac{1}{2}\mathbf{d}^T\nabla^2 L\left(\mathbf{x}^{(i)}\right)\mathbf{d} \qquad (4)$$

subject to

$$g_j\left(\mathbf{x}^{(i)}\right)+\nabla g_j\left(\mathbf{x}^{(i)}\right)^T\mathbf{d}\leq 0 \qquad (5)$$

$$h_j\left(\mathbf{x}^{(i)}\right)+\nabla h_j\left(\mathbf{x}^{(i)}\right)^T\mathbf{d}=0 \qquad (6)$$

where

$$\begin{aligned}\nabla^2 L\left(\mathbf{x}^{(i)}\right)=\nabla^2 f\left(\mathbf{x}^{(i)}\right)&\\ +\sum_j \mu_j^{(i)}\nabla^2 g_j\left(\mathbf{x}^{(i)}\right)&\\ +\sum_j \nu_j^{(i)}\nabla^2 h_j\left(\mathbf{x}^{(i)}\right)&\end{aligned} \qquad (7)$$

Throughout this work, the gradient vectors are assumed to be readily available while the Hessian matrices are updated using the quasi-Newton approximation. Even when the Hessian matrix required by the SQP is updated using quasi-Newton approximation, a super-linear rate of convergence is still achievable. However, it is well-known that algorithms in this class may not converge to the true global optimum. Any constrained optimization problems with nonlinear objective or constraint functions are in general intractable: it is impossible to design deterministic algorithms that could outperform the exhaustive search in assuring global convergence [10].

B. Stochastic Algorithms

A class of randomized algorithms capable of overcoming the limitation of deterministic optimization algorithms is the evolutionary computing. The most widely used technique in this class is the genetic algorithm (GA) [11]. Inspired by the natural inheritance of genes and the natural selection in the course of biological evolution [12], crossover, mutation, and survival of the fittest are at the heart of all GAs. The simplest form of the algorithm assumes only one population evolved from one generation to the next.

To deal with constraints, penalty functions are commonly used to penalize infeasible individuals. Different penalization schemes are available: death, static, dynamic, adaptive, and several others [13]. The ranking scheme proposed in [14] can also be employed to deal with constraints. Used throughout, the following three points summarize the scheme.

- The feasible solution is preferred to the infeasible one.
- Between any two feasible solutions, the one having better objective value is preferred.
- Between any two infeasible solutions, the one having less amount of constraint violations is preferred.

Over the past decade, research works on the evolutionary computing in the context of constrained optimization include Homomorphous Mapping [15], Stochastic Ranking [16], the Simple Multimembered Evolution Strategy (SMES) [17] that is known to have used the smallest number of fitness function evaluations (FFEs) so far, and several others (*e.g.* [18][19]). Although specially-designed operators may accelerate the search for the global optimum, it is agreeable that GA may suffer from excessively slow convergence trying to locate the optimum with sufficient precision because of its failure to exploit local information [20].

C. Memetic Algorithms

When designing an optimization procedure, two central yet competing goals must be handled: exploration vs. exploitation [21]. While the exploration provides some reliable estimates of the global optimum by surveying the entire search space, the exploitation enhances each estimate by concentrating the search efforts on its local neighborhood in order to produce a sufficiently accurate global optimum. Randomized algorithms accommodate the exploration while deterministic algorithms facilitate the exploitation. They will also be referred to as the global and local search methods hereafter. Hybridization of the two search methods has resulted in the memetic algorithm (MA). Motivated by Dawkins' notion of a meme (unit of cultural evolution capable of local refinements) [22], not only MA converges to high quality global optimum but also searches more efficiently [20]. The simplest variant of MA interleaves global with local search methods one after another and is known as the simple MA. It is intuitive that refining each individual in the population may not be the most efficient strategy. Local searches initiated from different locations may end up in the same local optimum and are thus wasted.

The choice of individuals that should undergo local refinements therefore becomes a critical issue in MA. In [3], a new paradigm of incorporating classification into the framework of optimization was first proposed to alleviate the aforementioned problem. The classification technique is used to distinguish feasible space from the infeasible one in the context of inequality-constrained optimization [4] or positive constraint space from the negative one for single equality constraint function [5]. The latter, however, assumes that the constraint function maps a set of design variables into an unbounded real value. For optimization problems whose constraint function exhibits restricted mapping into strictly non-negative or non-positive real values, alternative strategy is unquestionably necessary.

III. Problem Statement

Amidst the vast coverage of the optimization problems, this paper focuses on a special class of the equality-constrained problems which can be formulated without loss of generality as follows.

$$\min_{\mathbf{x}} f(\mathbf{x}) \tag{8}$$

subject to

$$h(\mathbf{x}) = 0 \tag{9}$$

where $h : \Re^n \to \Re\backslash\Re^-$. Note that maximizing $f(\mathbf{x})$ is equivalent to minimizing $-f(\mathbf{x})$ and solving $h(\mathbf{x}) = 0$ when $h : \Re^n \to \Re\backslash\Re^+$ is equivalent to finding $-h(\mathbf{x}) = 0$ when $h : \Re^n \to \Re\backslash\Re^-$, hence no loss of generality.

IV. Proposed Approach

The proposed approach will be presented in four subsections with the last summarizing the foregoing discussions from the first three in the form of a pseudo-code. In the first three subsections, the search for the global optimum will be discussed and the conditions around optimality shall be elaborated. This will be followed by discussing the neighborhood of any candidate solution from which optimality conditions can be perceived. Finally, the Support Vector Machine will be elaborated to make decision as to whether or not the candidate solution will undergo local refinement.

A. Global Optimum Search

In the context of the continuous constrained optimization, an optimum is reached when no further progress can be made by taking some infinitesimally small step therefrom such that the feasibility (satisfaction of all the constraints) is observed but the objective value is improved. Given some optimization problem with one and only one equality constraint, the possibly many optima (one of which shall be the global optimum) must therefore be situated at some particular point on the probably multi-dimensional curve defined by the set of feasible solution $X^0 = \{\mathbf{x} : h(\mathbf{x}) = 0\}$. In general, this set of feasible candidate solutions separates the infeasible individuals with positive constraint values from those with the negative ones. In other words, X^0 acts as a boundary between $X^+ = \{\mathbf{x} : h(\mathbf{x}) > 0\}$ and $X^- = \{\mathbf{x} : h(\mathbf{x}) < 0\}$. This should be intuitive as the constraint function is continuous. One has to go through a zero value for a transition from the positive to the negative ones. With this concept of the constraint value zero-crossing, it is proposed in [5] that local refinements can be selectively applied such that they converge to some local optimum more quickly as they are initialized with approximate solutions that are close enough to some optimal solution. The presence of previously evaluated candidate solutions with different signs of constraint values (positive and negative) within a neighborhood is used to create some classification model which can in turn be utilized to focus the search efforts on what is estimated to be the uncharted region of the optimization problem.

Hitherto, it has been assumed that the constraint function maps a candidate solution into unbounded real value. In practice, this may not always be the case. There are also problems whose constraint functions map a candidate solution into strictly non-negative (or non-positive) real value. For these problems, adopting the constraint value zero-crossing concept will obviously render the classification-assisted MA ineffective as it will behave like GA, except with longer time due to the overhead (neighborhood extraction and assessment) time. This is because the set X^0 no longer acts as a boundary between X^+ and X^-. Either X^+ or X^- will simply be an empty set. Unless the constraint function $h(\mathbf{x})$ is a constant mapping of candidate solution $\mathbf{x}$ to 0 (which implies that the problem is simply a continuous unconstrained optimization problem), then either the problem would not have any feasible candidate solution (the set X^0 is an empty set) or the feasible candidate solution set X^0 is the set of the global optimum point with respect to the constraint function h. The earlier is clearly not of our interest. The latter, however, is of great interest. It is actually the result of analyzing the topology of a constraint function which maps a candidate solution into strictly non-negative (or non-positive) real value.

Let us consider a constraint function with strictly non-negative mapping. Now, let us assume that we are starting from a candidate solution which the constraint function maps into a positive real value. As we are searching for a feasible candidate solution, we want to progress in the direction which minimizes the constraint value. Once we find a feasible candidate solution, taking infinitesimally small step in any direction should lead us into either another feasible candidate solution or an infeasible candidate solution with a positive constraint value as the mapping of the constraint function is strictly non-negative. This would also mean that traversing in the direction of any of the orthogonal coordinates of the design variables would increase the constraint value. Hence, we shall witness the change in the values of the partial derivatives of the constraint from downward to upward sloping as we pass through a feasible candidate solution $\mathbf{x} \in X^0$.

B. Neighborhood

As previously described, in the context of simple MAs, global and local searches are interleaved one after another allowing the individuals in the population to escape from the local optimum and concurrently converge to an optimum with sufficient precision. Given an individual of interest, what needs to be achieved is to be able to determine if it is situated at the insofar uncharted region of the search space of the optimization problem such that it is potentially worthy of local refinement. Should it be the case, only then the local refinement will be executed as it will converge more quickly if initialized with an estimate solution that is close enough to the optimum. This is beneficial in that the executed local refinement will require less function evaluations. Furthermore, with restricted budget in each local refinement, an initial estimate that is not sufficiently close to any optimum might end up nowhere near the optimum, either, wasting precious computational resources needed to perform the objective and constraint function evaluations. With an effective assessment strategy, the number of unsuccessful refinements will therefore be pushed as low as possible, hence saving further unnecessary functional evaluations. The benefit will surely be more obvious in the context of computationally expensive optimization problems whose objective function or constraint functions may take minutes to hours to execute for even just for single evaluation. Dealing with such problems, saving thousands or just hundreds of function evaluations could mean reduction of days or months of computation time. Moreover, what was initially unsolvable within reasonable time frame under the framework of simple MA could intuitively become more manageable should only the right choice of individuals will undergo local refinements. For that purpose, we need to assess the surrounding of that particular individual and apply the postulate from the previous section.

The examination of the surrounding of a particular individual of interest to assess its significance for local refinement may be achieved by probing either randomly or systematically some candidate solutions within a certain radius from the individual of interest. This, however, requires evaluations of the objective and constraint functions for each of the test probes in order to reveal a handful of meaningful information which will be of assistance to the assessment. Evaluating the test probes may incur too much overhead budget when dealing with computationally expensive optimization problems that necessitate numerous amounts of computational resources, both in term of time and memory. With minutes to hours of CPU time per evaluation, this approach is unmistakably undesirable. As an alternative, our intuition should tell us that such examination can also be achieved using the collection of past candidate solutions. The k nearest neighbors of the individual of interest can be extracted from this collection. In this paper, the Euclidean distance is used as the closeness measure between any two candidate solutions. The k neighbors would serve as the test probes that could assist the significance assessment of the individual of interest for a local refinement. As the collection comprises essentially previously evaluated candidate solutions, the neighborhood would already contain meaningful information about the surrounding of the particular individual of interest which has been delved into theretofore. Unlike in the previous approach, we may not have control over the radius of the so-called test probes but that also means one less parameter to fine tune. As the search for the global optimum progresses, the neighborhood represents the real topology of the problems more and more accurately. Adopting this latter strategy as opposed to the earlier one, learning the initially alien or hidden facts about the problem at hand based on data generated during the course of the optimization is made possible—a concept christened *Optinformatics*.

It is important to recall that the work presented in this paper addresses equality-constrained optimization with restricted (either strictly non-negative or strictly non-positive) constraint function mapping. Now, let us assume for the ease of discussion that the problem at hand has a strictly non-negative equality constraint function. (The foregoing discussion will also apply to problems with the strictly non-positive constraint function mapping by just flipping the negativity, *i.e.* negative becomes positive and vice versa). Clearly, the raw constraint values will be of no use as there exists no negative constraint value to distinguish from the positive ones. As discussed in the earlier section, however, the partial derivatives of the constraint function shall consist of negative and positive real values nearby a feasible candidate solution. As zero is the globally lowest value in the strictly non-negative constraint function, the solution to the equality constraint $h(\mathbf{x}) = 0$ would coincide with the global optimum of the constraint function h. At optimum, all partial derivatives of a function should vanish. This means $\partial h(\mathbf{x})/\partial x_i = 0$ for all possible i. Examining the values of all of the partial derivatives of the constraint within a neighborhood will thus allow us to reveal if we are sufficiently close to a feasible candidate solution $\mathbf{x}$ for which $h(\mathbf{x}) = 0$. Presence of either all-positive or all-negative values of the derivative is an indication of the absence of any feasible solution within the neighborhood. On the other hand, presence of both negative and positive values of derivative from the neighborhood would allows us to estimate the insofar unexamined region near the equalities $\partial h(\mathbf{x})/\partial x_i = 0$ (or equivalently the equality $h(\mathbf{x}) = 0$).

C. Support Vector Machines

Given a neighborhood whose members are of different signs with respect to the values of one of the constraint derivatives, a hypersphere in which a feasible solution to the optimization problem may be located has therefore been identified. The subsequent thing to achieve is then to be able to predict the insofar unexamined region within the hypersphere. Should the individual of interest for which the significance assessment for local refinement is carried out be located in the estimated-to-be-unfathomed region of the hypersphere, local refinement shall therefore be really performed in order to reveal more information behind the previously unexamined region. For this purpose, we must first be able to distinguish the space of positive partial derivative values from that of the negative ones within the hypersphere.

As the constraint functions of the problems addressed in this paper are continuous and differentiable, all partial derivatives of the constraint functions, therefore, must also be continuous. It is then reasonable to assume that there will not be any abrupt changes in the partial derivative values such that when the neighborhood is sufficiently localized, the information captured from the neighbors will be enough to roughly separate the hypersphere of the neighborhood into a subspace of positive derivative values at one side and a subspace of negative derivative values at the other side. In between these two subspaces, there lies a subspace where nobody can ever be very sure of what would be the values of the derivatives until further investigations are performed. At this unidentified subspace, too, the boundary that distinguishes positive derivative values from the negative ones should reside. Therefore, local refinement initiated with any individual located in this subspace would be of significance importance. Our goal is hence to be able to model this alien subspace.

Support Vector Machine (SVM) [23] is a machine-learning technique initially proposed as a two-class linear classifier. It is well-known for its capability to maximize the geometric margin between the two classes while simultaneously minimize the classification errors. Being linear classifier, the SVM therefore has a decision function of the from $D(\mathbf{x}) = \mathbf{w}\cdot\mathbf{x} + w_0$. Upon the provision of some training data instances $(\mathbf{x}_i, y_i)$ where $\mathbf{x}_i \in \Re$, $y_i \in \{-1, +1\}$, and $i = 1, 2, \ldots, k$, the SVM will attempt to find an appropriate set of coefficients $(\mathbf{w}, w_0)$ that will altogether define the decision function $D(\mathbf{x})$ such that $y_i D(\mathbf{x}_i) \geq 1$. In other words, the positive training data instances will have decision values of at least +1 while the negative ones will have decision values of at most –1. As there are only finite number of training data instances, there must then exists a subspace that separates the positive training data from the negative ones but reveals no clear-cut conclusion regarding the class of any previously unseen data should they be situated thereat. In this alien subspace, the SVM decision values are between –1 and +1. This region is known as the margin-of-separation whose width is given by $2/\|\mathbf{w}\|$. To maximize the margin, therefore, the SVM needs to minimize

$$\frac{1}{2}\|\mathbf{w}\|^2 \tag{10}$$

subject to

$$y_i\left[\left(\mathbf{w}\cdot\mathbf{x}_i\right)+w_0\right]\geq 1 \tag{11}$$

for all $i = 1, 2, \ldots, k$, in which the factor of 1/2 is introduced only for mathematical convenience. In its dual form, the SVM problem formulation above is equivalent to finding the set of alpha values that maximizes the following quadratic program where i and j are indices of the training data. It should be noted that each α corresponds to a training data instance.

$$\sum_{i=1}^{k}\alpha_i - \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}\alpha_i\alpha_j y_i y_j\left(\mathbf{x}_i\cdot\mathbf{x}_j\right) \tag{12}$$

subject to

$$\sum_{i=1}^{k}\alpha_i y_i = 0 \tag{13}$$

$$\forall i\ \alpha_i \geq 0 \tag{14}$$

Collection of the training data instances with non-zero positive α values defines the support vectors. They support the decision surfaces $D(\mathbf{x}) = +1$ or $D(\mathbf{x}) = -1$ depending on which class they belong to. Based on these support vectors, the weight vector $\mathbf{w}$ and bias w_0 can then be computed using the following formulas in which SV is the set of support vectors indices.

$$\mathbf{w} = \sum_{i=1}^{k} \alpha_i y_i \mathbf{x}_i = \sum_{i \in SV} \alpha_i y_i \mathbf{x}_i \tag{15}$$

$$w_0 = \frac{1}{|SV|} \sum_{i \in SV} \left(y_i - \sum_{j=1}^{k} \alpha_j y_j \left(\mathbf{x}_j \cdot \mathbf{x}_i \right) \right) \tag{16}$$

As a classification technique, the SVM nicely fits the concept of distinguishing the continuous landscape of any of the partial derivatives of the constraint function into three subspaces: the area of positive derivative, the area of negative derivative, and the area that is yet-to-be-investigated. Given a neighborhood $\aleph$ of any particular individual $\mathbf{x}$ of interest, *i.e.* $\aleph(\mathbf{x}) = \{\mathbf{x}_j : \mathbf{x}_j$ is the k nearest neighbors of $\mathbf{x}\}$, the neighbors $\mathbf{x}_j$, $j = 1, 2, \ldots$, k, can then serve as training data for the SVM with $y_j = \text{sign}(\partial h(\mathbf{x}_j)/\partial x_i)$ should y_j is neither all-positive nor all-negative. After the SVM model is constructed, the individual $\mathbf{x}$ should then be tested against the model to see its location in the approximated structure of the neighborhood hypersphere. As there as a number of partial derivatives, depending on the dimensionality of the problem, multiple SVM models are therefore required. Should the individual $\mathbf{x}$ be located in the intersection of the margin-of-separations of every SVM model constructed, then it would be significant for local refinement.

D. Complete Framework

Initialize a population
Evaluate the population
while no stopping criteria have been fulfilled **do**
  for each individual $\mathbf{x}$ in the population **do**
    Identify the neighborhood $\aleph(\mathbf{x})$
    refine = true
    for i = 1 to problem size
      if $y_j = \text{sign}(\partial h(\mathbf{x}_j)/\partial x_i)$ is not all-positive or all-negative **then**
        Train SVM based on $\aleph(\mathbf{x})$ and y_j to obtain decision function $D(\cdot)$
        if $|D(\mathbf{x})| > 1$ **then**
          refine = false
          exit for
        end if
      end if
    end for
    if *refine* **then**
      Refine $\mathbf{x}$ using local search
    end if
  end for
  Evolve the population through crossover, mutation, and elitism
  Evaluate the population

Figure 1. Classification-assisted Memetic Algorithm for Equality-constrained Optimization Problems with Restricted Constraint Function Mapping

To summarize the foregoing discussions of the proposed approach, a complete framework in the form of a pseudo-code is presented in the Fig. 1.

V. RESULTS AND DISCUSSIONS

Using GA as the global search and SQP as the local search method, an empirical study was carried out with a population size of 100 individuals and a maximum of $10n$ fitness function evaluations (FFEs) for each individual refinement with n being the dimensionality of the problem. A neighborhood size of $2n$ was also assumed for the proposed method. Experimented with is the modified version of the benchmark problem **g03** [24] for $n = 2, \ldots, 6$; the original functions of which can be found in [25]. The objective function is not altered while the constraint function is shifted and offset to mimic the characteristic of the constraint function addressed herein. The explicit formulation of the constraint function is $h(\mathbf{x}) = \sum_i (x_i - 1/\sqrt{n})^2$. This is, as the matter of fact, a more difficult problem to solve compared to the original one as the feasible space comprises only a single point. The number of the dimensionality is chosen for the sake of demonstrating the following three aspects: evaluation count, generation count, and optimization time tabulated respectively in Table I, II, and III.

A. Evaluation Count

TABLE I. AVERAGE NUMBER OF FITNESS FUNCTION EVALUATIONS REQUIRED TO SOLVE THE BENCHMARK PROBLEM OF DIFFERENT SIZES

n	Simple MA	Classification-assisted MA
2	11,397 ± 5,131	2,150 ± 256
3	24,756 ± 10,418	3,724 ± 382
4	28,462 ± 16,240	5,171 ± 723
5	31,927 ± 20,077	8,716 ± 1,418
6	31,934 ± 20,742	13,946 ± 3,245

From Table I, it can be seen that reduction up to ~80% in terms of evaluation count is achievable when using the proposed method as opposed to when using the simple MA. Recall that an equality-constrained optimization problem has minute feasible space. Using the simple MA where refinements are performed for every individual in the population, many of them may not actually bring out the individuals anywhere near some optimum due to limited budget of each local refinement (it is $10n$ FFEs at maximum in the case of this empirical study as mentioned above). The refinements are thus wasted. In other words, they are not actually necessary in the first place. By refining only individuals predicted to be located near a feasible space through guided assessments based on the constraint derivative values of its neighbors, the proposed method has indeed been successful in reducing significantly the amount of FFEs required to find the global optimum. In the context of computationally expensive optimization, such reduction would unquestionably be beneficial.

B. Generation Count

TABLE II. AVERAGE NUMBER OF GENERATIONS REQUIRED TO SOLVE THE BENCHMARK PROBLEM OF DIFFERENT SIZES

n	**Simple MA**	**Classification-assisted MA**
2	8.3 ± 3.0	21.5 ± 2.5
3	10.8 ± 3.9	36.3 ± 4.1
4	9.3 ± 4.7	51.3 ± 7.9
5	8.8 ± 4.9	91.2 ± 15.1
6	8.1 ± 4.8	149.8 ± 34.9

From Table II, it is clear that the proposed method allows more evolution in the course of the optimization compared to the simple MA. This is arguably beneficial as less evolution may lead to premature convergence. Local refinement for every individual in the population in the context of the simple MA may cause several individuals to converge to the same local optimum. When too many individuals actually converge to the same local optimum, diversity in the population may decrease significantly. When this happens, even the global search may face difficulty in escaping the local optimum: the problem of *premature convergence*. The proposed method, on the other hand, arguably allows diversity to be maintained as only potential individuals will undergo local refinements. Hence, the proposed approach yields a better guarantee of locating the global optimum.

C. Optimization Time

TABLE III. AVERAGE CPU TIME (IN SECONDS) REQUIRED TO SOLVE THE BENCHMARK PROBLEM OF DIFFERENT SIZES

n	**Simple MA**	**Classification-assisted MA**
2	18.7 ± 7.4	3.1 ± 0.4
3	47.2 ± 23.5	10.8 ± 1.4
4	69.5 ± 52.1	26.1 ± 4.7
5	78.4 ± 58.4	70.3 ± 12.5
6	84.7 ± 74.6	159.7 ± 47.0

In Table III, the average CPU time taken to perform the optimization is tabulated. When the problem dimensionality is sufficiently small ($n \leq 5$), it can be seen from the table that there is savings in term of CPU time even in the context of computationally-inexpensive optimization problems, *i.e.* the benchmark problem considered herein. This is because both the SVM and the SQP need to solve a quadratic program, which is the most expensive component in MA using GA and SQP. The reduction in term of number of FFEs, however, outweighs the overheads including that due to the SVM model constructions. For higher dimensionalities, nonetheless, the reduction in terms of number of FFEs can no longer compensates the time required for the overheads as there are more SVM models to construct as the dimensionality increases. This is witnessed when $n = 6$. Such phenomenon, however, will definitely not be the case in the context of computationally expensive optimization problems as the evaluations of the objective and constraint functions of the problems will be the most expensive components. Hence, reduction in term of number of FFEs will undeniably become quite significant in reducing the CPU time.

VI. CONCLUSIONS AND FUTURE WORKS

Utilizing signs of the raw constraint values within some neighborhood for the supervised learning by Support Vector Machines (SVMs), one earlier attempt [5] has been successful to enhance the efficiency of Memetic Algorithms (MAs) when solving optimization problems with single equality constraint and unbounded constraint function mapping. This was done by sensing the zero-crossing area of the constraint values. In this paper, the issue of solving similar optimization problems with restricted constraint function mapping is addressed. As the constraint values are either non-negative or non-positive, the partial derivative values of the constraint are used as the source of information to sense the zero-touching area of the constraint values. Empirical studies further demonstrate the efficacy of the proposed approach.

With up to around 80% saving in terms of evaluation count, using the proposed approach would undoubtedly be beneficial in the context of computationally expensive optimization problems as compared to using the simple MA. Reduction of thousands of FFEs easily translates into shorter optimization time at the scale of days or months when single evaluation of the objective or constraint functions requires minutes to hours to execute. In the context of computationally inexpensive problems, such as the benchmark used in this study, the overhead incurred by the SVM modeling may be too high as discussed in the previous section. It would therefore be interesting to study the use of less expensive classifiers for the modeling in the future. It may also be interesting to attempt applying the proposed approach to real-time system when using less expensive classifier, such as [26].

REFERENCES

[1] Bazaraa, M.S., H.D. Sherali, and C.M. Shetty, *Nonlinear Programming: Theory and Algorithms*, Wiley-Interscience, 2006.

[2] Bambha, N.K., et al., "Systematic Integration of Parameterized Local Search into Evolutionary Algorithms," *IEEE Transactions on Evolutionary Computation*, 8(2), pp. 137–155, 2004.

[3] Handoko, S.D., C.K. Kwoh, and Y.S. Ong, "Using Classification for Constrained Memetic Algorithm: A New Paradigm," in *Proceedings of the 2008 IEEE International Conference on Systems, Man and Cybernetics*, 2008.

[4] Handoko, S.D., C.K. Kwoh, and Y.S. Ong, "Feasibility Structure Modeling: An Effective Chaperon for Constrained Memetic Algorithms," *IEEE Transactions on Evolutionary Computation*, 14(5), pp. 740–758, 2010.

[5] Handoko, S.D., C.K. Kwoh, and Y.S. Ong, "Classification-assisted Memetic Algorithms for Equality-Constrained Optimization Problems," in *Lecture Notes in Artificial Intelligence*, 5866, pp. 391–400, 2009.

[6] Zoutendijk, G., *Methods of Feasible Directions*, Elsevier, 1960.

[7] Baker, T.E. and L.S. Lasdon, "Successive Linear Programming at Exxon," *Management Science*, 31(3), pp. 264–274, 1985.

[8] Zhang, J.Z., N.H. Kim, and L.S. Lasdon, ”An Improved Successive Linear Programming Algorithm,” *Management Science*, 31(10), pp. 1312–1331, 1985.

[9] Wilson, R.B., ”A Simplicial Algorithm for Convex Programming,” Ph.D. Thesis, Harvard University, 1963.

[10] Wright, S.J., ”Nonlinear and Semidefinite Programming,” in *Proceedings of Symposia in Applied Mathematics*, 61, pp. 115–138, 2004.

[11] Holland, J.H., *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, 1975.

[12] Darwin, C., *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*, John Murray, 1859.

[13] Coello, C.A.C., ”Theoretical and Numerical Constraint-Handling Techniques used with Evolutionary Algorithms: A Survey of the State of the Art,” *Computer Methods in Applied Mechanics and Engineering*, 191(11–12), pp. 1245–1287, 2002.

[14] Deb, K., ”An Efficient Constraint Handling Method for Genetic Algorithms,” *Computer Methods in Applied Mechanics and Engineering*, 186(2–4), pp. 311–338, 2000.

[15] Koziel, S. and Z. Michalewicz, ”Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization,” *Evolutionary Computation*, 7(1), pp. 19–44, 1999.

[16] Runarsson, T.P. and Y. Xin, ”Stochastic Ranking for Constrained Evolutionary Optimization,” *IEEE Transactions on Evolutionary Computation*, 4(3), pp. 284–294, 2000.

[17] Mezura-Montes, E. and C.A.C. Coello, ”A Simple Multi-Membered Evolution Strategy to Solve Constrained Optimization Problems,” *IEEE Transactions on Evolutionary Computation*, 9(1), pp. 1–17, 2005.

[18] Hamida, S.B. and M. Schoenauer, ”ASCHEA: New Results Using Adaptive Segregational Constraint Handling,” in *Proceedings of the 2002 Congress on Evolutionary Computation*, pp. 884–889, 2002.

[19] Elfeky, E.Z., R.A. Sarker, and D.L. Essam, ”A Simple Ranking and Selection for Constrained Evolutionary Optimization,” in *Lecture Notes in Computer Science*, 4247, Springer-Verlag, 2006.

[20] Tang, J., M.H. Lim, and Y.S. Ong, ”Diversity-Adaptive Parallel Memetic Algorithm for Solving Large Scale Combinatorial Optimization Problems,” *Soft Computing: A Fusion of Foundations, Methodologies, and Applications*, 11(9), pp. 873–888, 2007.

[21] Torn, A. and A. Zilinskas, *Global Optimization*, Springer-Verlag, 1989.

[22] Dawkins, R., *The Selfish Gene*, Oxford University Press, 1976.

[23] Vapnik, V.N., *The Nature of Statistical Learning Theory*, Springer, 1995.

[24] Liang, J.J., et al., *Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization*, Technical Report, 2006.

[25] Michalewicz, Z., G. Nazhiyath, and M. Michalewicz. "A Note on Usefulness of Geometrical Crossover for Numerical Optimization Problems," in *Proceedings of the 5th Annual Conference on Evolutionary Programming*, pp. 305-312, 1996.

[26] Tsang, I.W., A. Kocsor, and J.T. Kwok. "Simpler Core Vector Machines with Enclosing Balls," in *Proceedings of the 24th International Conference on Machine Learning*, 2007.