Feasibility Structure Modeling: An Effective Chaperone for Constrained Memetic Algorithms

S.D. Handoko, C.K. Kwoh, and Y.S. Ong

Abstract

An important issue in designing memetic algorithms (MAs) is the choice of solutions in the population for local refinements, which becomes particularly crucial when solving computationally expensive problems. With single evaluation of the objective/constraint functions necessitating tremendous computational power and time, it is highly desirable to be able to focus search efforts on the regions where the global optimum is potentially located so as not to waste too many function evaluations. For constrained optimization, the global optimum must either be located at the trough of some feasible basin or some particular point along the feasibility boundary. Presented in this paper is an instance of *optinformatics* [1] where new concept of modeling the feasibility structure of inequality-constrained optimization problems—dubbed the *Feasibility Structure Modeling* (FSM)—is proposed to perform geometrical predictions of the locations of candidate solutions in the solution space: deep inside any infeasible region, nearby any feasibility boundary, or deep inside any feasible region. This knowledge may be unknown prior to executing MA but it can be mined as the search for the global optimum progresses. As more solutions are generated and subsequently stored in the database, the feasibility structure can thus be approximated more accurately. As an integral part, a new paradigm of incorporating the classification—rather than the regression—into the framework of MAs is introduced, allowing the MAs to estimate the feasibility boundary such that effective assessments of whether or not the candidate solutions should experience local refinements can be made. This eventually helps preventing the unnecessary refinements and consequently reducing the number of function evaluations required to reach the global optimum.

1 Introduction

Increasing success of the memetic algorithms (MAs) to perform more efficiently compared to their conventional counterparts [2, 3] in the search for optimal solutions has driven the focus of current and future research works towards solving computationally-expensive optimization problems. Characterized by the enormous amount of computational power and time required to perform even a single evaluation of the objective and/or constraint functions, only limited amount of fitness function evaluations (FFEs) would in general be affordable for problems in this category. Even if the gargantuan computational cost to completely solve these problems could possibly be made available, smaller number of FFEs is always desirable. Reductions of tens to hundreds of FFEs would translate easily to savings of hours to days of computational time when single FFE took minutes or even hours to complete. Many practical optimization problems belong to this problem category. An inspiring example could be the rational design of vaccines [4] for which calculation of the potential energy to minimize involves many atoms. Depending on the fidelity of the model employed, a single FFE would necessitate minutes to hours of computational time to complete. For such an expensive problem, domain knowledge incorporation shall undeniably be beneficial [5].

A canonical MA that simply interleaves the global search through stochastic optimization algorithm and the local search through deterministic optimization method one after another, renowned as the simple MA, may not be the most efficient approach for locating the optimal solutions. Local search on each candidate solution in the population incurs a large number of FFEs. Hence, choice of the candidate solutions the local search should refine is an important concern in designing MA [7]. Fitness- and distribution-based strategies were studied in [8, 9] for adapting the probability of executing local refinements by means of simple heuristics, *e.g.* best in the population. In [10], formalized probabilistic method was proposed. This becomes increasingly essential in the context of computationally-expensive problems.

For an inequality-constrained optimization problem, it is well-established that its optimal solutions may be located at the trough of some feasible basin—resembling the unconstrained optimization problem—or at some particular point along the feasibility boundary. Obviously, one of the possibly many optimal solutions ought to be the global optimum. If the feasibility structure of the problem could be known a priori, the search efforts may be easily focused on some promising regions so as to be able to find the global optimum within minimum amount of FFEs possible. Unfortunately, this is usually not the case. While it is possible to examine analytically the feasibility structures of problems with low dimensionality, it could have been rather intractable to do so for higher-dimensional problems. Fortunately, it is not impossible for the MA—as it evolves—to grasp local feasibility structure about some candidate solution, *i.e.* whether the solution seems to be located (1) deep inside a feasible search space, (2) deep inside an infeasible search space, or (3) near some feasibility boundary. This information can hence be used to determine whether the solution is significant for a local refinement recalling that the optimal solutions can possibly be located only at two different locations. Exploiting the neighborhood of the candidate solution based on past solutions evaluated previously and employing machine-learning (classification) technique as required, the proposed framework—denoted as *Feasibility Structure Modeling* (FSM)—paves a way to mine the knowledge about the local feasibility structure of the problem being optimized such that it is possible to make decisions regarding the necessary actions to perform at certain point during the optimization based on the evaluated candidate solutions hitherto. FSM is indeed a form of *optinformatics* [1]—the specialization of informatics for the processing of the data generated in optimization in order to extract possibly implicit and potentially useful information and knowledge, which could be helpful for understanding search mechanism of the solver, guiding/improving search, and/or revealing undisclosed insights to the problem structure. In FSM, feasibility structure is more and more accurately modeled as the optimization advances focusing the search effort more and more effectively on the promising regions where the optimal solutions may reside.

According to No-Free-Lunch theorem: "algorithm performs only as well as the knowledge concerning the cost function put into it" [11]. It is thus highlighted by the theorem the need to incorporate domain knowledge about the problem to solve to attain good performance [5]. This implies that incorporation of the knowledge mining discussed above is indispensable for achieving good MA performance.

In what follows, this paper is organized in a way that the problem addressed is explicitly stated in Section 2 and previous attempts of solving such problem are discussed in Section 3. Supporting theories for modeling the feasibility structure of the problem are then elaborated in Section 4. Empirical results are presented in Section 5 (thirteen benchmark problems) and Section 6 (a real-world application). Finally, Section 7 concludes this paper and presents few future works.

2 Problem Statement

This research work addresses numerical inequality-constrained optimization problems, which aim at identifying the vector $\mathbf{x}$ of n continuous independent variables that minimizes

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

subject to

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

where $\mathbf{x} \in \Re^n$ is the *candidate solution* while $f : \Re^n \rightarrow \Re$ and $\mathbf{g} : \Re^n \rightarrow \Re^{n_g}$ are, respectively, the *objective* function and the *inequality constraint* functions; all of which are assumed to be continuous and numerically differentiable. In addition, there could exist bound constraints in the form of $\underline{\mathbf{x}} \leq \mathbf{x} \leq \overline{\mathbf{x}}$ in which $\underline{\mathbf{x}}$ is the *lower bound* and $\overline{\mathbf{x}}$ is the *upper bound*. This research is thus focused on the numerical optimization of the general inequality-constrained problems, a subset of the renowned nonlinear programming.

3 Previous Works

3.1 Deterministic Algorithm

A class of deterministic algorithms called the *methods of feasible directions* [12] has long been developed to handle nonlinear program. Methods in this class proceed in an iterative manner from one feasible solution to another. In each single iteration, a direction-finding subproblem is solved; the solution of which is then used to direct a line search along which the search for the global optimum should move. Making use of the second-order functional approximations so as to achieve a quadratic convergence rate is the *sequential quadratic programming* (SQP) [13, 14, 15], a widely-used nonlinear programming solver.

By the Newton's method, the SQP solves directly the KKT optimality conditions [16, 17] for the local minimizers; one of which must be the global optimum of the nonlinear program. For every single iteration of the Newton's method, there exists an accompanying subproblem which happens to be minimization of the quadratic approximation to Lagrangian function of the nonlinear program subject to the set of linear approximations to all constraint functions. Given the iterate $(\mathbf{x}^{(k)}, \mu^{(k)})$ where $\mu^{(k)} \geq \mathbf{0}$ are the estimates of the Lagrange multipliers for the inequality constraints, the quadratic program below is solved at every major iteration of the SQP for a feasible direction $\mathbf{d}$.

$$f(\mathbf{x}^{(k)}) + \nabla f(\mathbf{x}^{(k)})^T \mathbf{d}^{(k)} + \frac{1}{2}[\mathbf{d}^{(k)}]^T \nabla^2 L(\mathbf{x}^{(k)}) \mathbf{d}^{(k)} \tag{3}$$

subject to

$$g_i(\mathbf{x}^{(k)}) + \nabla g_i(\mathbf{x}^{(k)})^T \mathbf{d}^{(k)} \leq 0 \qquad\qquad i = 1, \ldots, n_g \tag{4}$$

where

$$\nabla^2 L(\mathbf{x}^{(k)}) = \nabla^2 f(\mathbf{x}^{(k)}) + \sum_{i=1}^{n_g} \mu_i^{(k)} \nabla^2 g_i(\mathbf{x}^{(k)}) \tag{5}$$

While a quadratic rate of convergence can be achieved, convergence to the local optimum is an inherent characteristic of deterministic optimization algorithms. Hence, it is impossible to devise any deterministic algorithm that could outperform an exhaustive search in assuring global convergence for nonlinear programs in the context of global optimization are generally intractable [18].

3.2 Stochastic Algorithm

As a remedy to the limitation of deterministic optimization techniques, stochastic algorithms such as the *genetic algorithm* (GA) [19, 20] is often used. Inspired by the natural inheritance of genetic materials as well as the natural selections in the course of biological evolution [21], GA can escape from local optima and converge to the global optimum given sufficiently long evolution, owing to its crossover and mutation operators. Being the unconstrained optimizer, GA then needs the constraint-handling techniques [22] to deal with constrained optimization problems. Previously proposed approaches include some penalty functions [23, 24, 25], repair method [26], diversity mechanism [27], stochastic [28] and several other [29, 30, 31] rankings. Summarized in the following three points, the ranking scheme in [29] is used throughout this research work.

- A feasible solution is preferred to the infeasible one.
- Between two feasible solutions, the one with the better objective value is preferred.
- Between two infeasible solutions, the one with the lesser amount of constraint violation is preferred.

Despite the success in converging to the true global optimum, it is a well-known fact that stochastic algorithms may suffer from an excessively slow convergence attempting to identify the global optimum with sufficient precision due to the failure in exploiting local information of the search space.

3.3 Hybrid Algorithm

Trend of the SQP as a deterministic algorithm to identify only a local rather than the global optimum and the adversity of GA as a stochastic algorithm to locate with sufficient precision the global optimum are effective indications that each of them lacks one of the two main yet competing goals in the design of an optimization procedure: *exploration* vs. *exploitation* [32]. While exploration produces reliable estimates of the global optimum, the exploitation refines each estimate so as to be able to identify the global optimum with sufficiently high precision. The hybridization of GA with the SQP has thus become one feasible alternative to achieving the balance between these two goals.

Inspired by the meme [33]—a cultural entity capable of refinement—such hybridization is hence referred to as the *memetic algorithm* (MA) [34, 35, 36, 37, 38]. Simplest incorporation of the exploitation to the exploration is intuitively the interleaving of GA and SQP one after another as portrayed by Algorithm 1. The simple MA carries out local refinement on each of the candidate solutions in the current population prior to evolving them to establish the next population. Consequently, not only is it able to accurately pinpoint the true global optimum but also the simple MA searches more efficiently than its counterparts [2]: a multi-start SQP and the pure GA.

In a single iteration, the FFE requirement of the simple MA is obviously far greater than that of the simple GA. Should each candidate solution be allotted a maximum of ℓ FFEs per local refinement and let there be m candidate solutions in the population, up to $m \times \ell$ FFEs will be required by the simple MA in a single generation while only m FFEs will be required by the simple GA under the same assumptions. Because of the high computational demands in each of the local refinements, approximation of the objective and the constraint landscape of an optimization problem [39, 40, 41] has become the alternative to reduce the requirement of the FFEs.

Algorithm 1 Simple Memetic Algorithm

```
Initialize a population of candidate solutions
Evaluate the fitness of each candidate solution
for each candidate solution in the population do
    Refine the candidate solution locally
end for
while no stopping criteria have been fulfilled do
    Evolve the current population by one generation
    Evaluate the fitness of each candidate solution
    for each candidate solution in the population do
        Refine the candidate solution locally
    end for
end while
```

4 Proposed Approach

Before elaborating the proposed approach in detail, it is important to note that the intended use of the methodology is the optimization of some computationally-expensive problems. For evaluations of the objective and/or constraint functions of the problems demand tremendous computational resources, re-evaluations of candidate solutions during the course of searching for the global optimum are clearly undesirable. Therefore, a database of previously evaluated candidate solutions needs to be maintained. By recording the coordinates and the evaluation outcomes of each newly encountered candidate solution in a tabular form, a simple database has been established. Although this brings about additional house-keeping activities, such as the recording of information for each newly evaluated candidate solution and the inquiring of the database prior to evaluation of every candidate solution, not only shall the re-evaluations of candidate solutions be avoided but also neighborhood identification of a candidate solution has been made possible; the latter of which is indeed an important element for the proposed approach.

4.1 Neighborhood Structure

Given some particular candidate solution $\mathbf{x}$, the neighborhood is contributed by k previously evaluated candidate solutions sustaining k shortest distances to $\mathbf{x}$. By representing the set of the candidate solutions in the database as $\mathcal{DB}$ and the set of the k neighbors of $\mathbf{x}$ as $\mathcal{N}$, it is logical that the inequality

$$dist(\mathbf{x}, \mathbf{x}^i) \geq dist(\mathbf{x}, \mathbf{x}^j) \tag{6}$$

in which $\mathbf{x}^i \in \mathcal{DB} \backslash \{\mathcal{N} \cup \{\mathbf{x}\}\}$ and $\mathbf{x}^j \in \mathcal{N}$ must be observed for a distance measure $dist(\cdot, \cdot)$. Used throughout this work is the Euclidean distance which measures the sparseness between any pairs of candidate solutions. In the n-dimensional space, it is given by

$$dist(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}) = \sqrt{\sum_{i=1}^{n} \left(x_i^{(1)} - x_i^{(2)} \right)^2} \tag{7}$$

where $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$ are any two arbitrary n-dimensional vectors having $x_i^{(1)}$ and $x_i^{(2)}$ as their respective i-th element.

The neighborhood $\mathcal{N}$ of some candidate solution $\mathbf{x}$ for some integer k can mathematically be written as

$$\mathcal{N}(\mathbf{x}|k) = \{\mathbf{x}^i : \mathbf{x}^i \neq \mathbf{x}, \mathbf{x}^i \in \mathcal{DB}, rank(dist(\mathbf{x}, \mathbf{x}^i)) \leq k\} \tag{8}$$

in which the function $rank(\cdot)$ determines the relative rankings of all the evaluated candidate solutions stored in the database in the ascending order of distances to the candidate solution $\mathbf{x}$ against which the neighborhood is defined. Implicit information which can be derived from a neighborhood is the feasibility of each of the members. Depending on its constraint values, a neighbor can be a feasible or an infeasible candidate solution. Based on this, three types of neighborhood may be encountered in the course of searching for the global optimum: *feasible*, *infeasible*, and *mixed* neighborhood. Some illustrative examples are shown in Figure 1 where two feasible, two infeasible, and a mixed neighborhood are respectively denoted as **A** and **B**, **C** and **D**, and **E**.

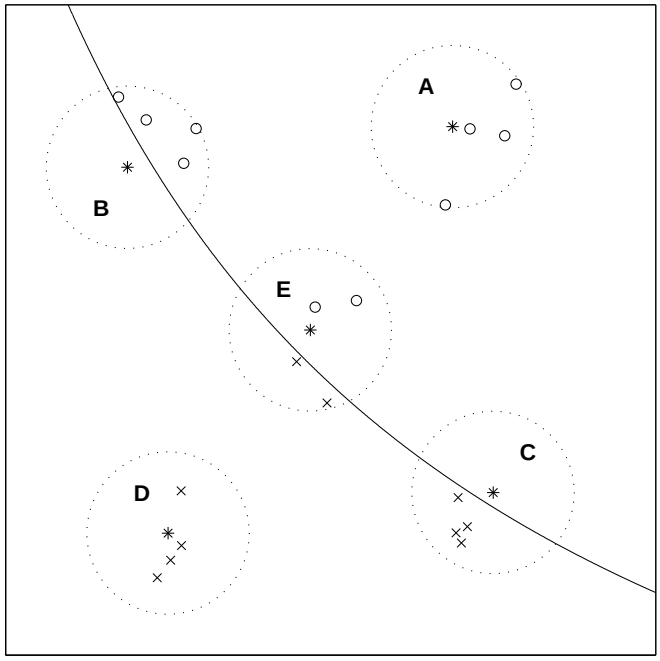

Figure 1: Neighborhood Structure
$*$: candidate solution with respect to which the neighborhood is defined.
$\circ$: feasible candidate solution. $\times$: infeasible candidate solution.

4.1.1 Feasible Neighborhood

A feasible neighborhood is composed entirely of the feasible candidate solutions as suggested by the name. Direct intuition should advise that a feasible neighborhood signifies an entirely feasible search space within the locality of some candidate solution $\mathbf{x}$ with reference to which the feasible neighborhood has been defined. More careful investigation, however, reveals that the feasible neighborhood could also represent a partially feasible search space, depending on whether the particular candidate solution $\mathbf{x}$ is feasible or infeasible.

For a feasible candidate solution $\mathbf{x}$, the search space encompassed by the neighborhood is seemingly unconstrained (*cf.* Figure 1: neighborhood $\mathbf{A}$). Unless it is known beforehand that the problem being optimized is characterized by the global optimum at some particular point along the feasibility boundary, it will only be trustworthy to perform a local refinement on $\mathbf{x}$ as the global optimum could have also been located at the trough of the basin of $\mathbf{x}$.

On the contrary, an infeasible candidate solution $\mathbf{x}$ denotes that the search space covered by the feasible neighborhood is undoubtedly constrained (*cf.* Figure 1: neighborhood **B**). An existence of the feasibility boundary between $\mathbf{x}$ and its neighbors is therefore implied. Unless it is known beforehand that the problem being solved is characterized by the global optimum at the trough of some feasible basin, it will only be dependable to perform a local refinement on $\mathbf{x}$ as the global optimum could have also been located along the feasibility boundary.

4.1.2 Infeasible Neighborhood

In contrast to the previous type of neighborhood, an infeasible neighborhood is composed of the infeasible candidate solutions entirely as pinpointed by the name. Direct intuition should thus advise that an infeasible neighborhood denotes an entirely infeasible search space within the locality of some candidate solution $\mathbf{x}$ with reference to which the infeasible neighborhood has been defined. A cautious investigation, however, reveals that the infeasible neighborhood might similarly signify a partially feasible search space, depending on whether the particular candidate solution $\mathbf{x}$ is feasible or infeasible.

For a feasible candidate solution $\mathbf{x}$, the search space encompassed by the neighborhood is unmistakably constrained. The feasibility boundary must be present between $\mathbf{x}$ and all of its neighbors (*cf.* Figure 1: neighborhood **C**). Sometimes, this may mean the candidate solution $\mathbf{x}$ lies on a disjointed—previously unexplored—feasible search space. For the global optimum could have been located inside the particular disjointed feasible space, it will only be reliable to perform a local refinement on $\mathbf{x}$ regardless of whether or not there is a priori knowledge of the location of the global optimum.

Conversely, it will be wise not to perform any local refinement on an infeasible candidate solution $\mathbf{x}$ until the global search can indicate that there is indeed a disjointed feasible space in the vicinity of $\mathbf{x}$ since an infeasible $\mathbf{x}$ signifies a seemingly infeasible search space as far as the locality of $\mathbf{x}$ is concerned (*cf.* Figure 1: neighborhood **D**).

4.1.3 Mixed Neighborhood

Finally, a mixed neighborhood, as implied by the name, comprises a mixture of both feasible and infeasible candidate solutions (*cf.* Figure 1: neighborhood **E**). Some feasibility boundary must be present in this form of neighborhood regardless of whether the candidate solution $\mathbf{x}$ against which the neighborhood has been defined is feasible or infeasible. Upon an encounter with mixed neighborhood, further investigations are necessary so as to determine if $\mathbf{x}$ lies on some previously unexplored region nearby the feasibility boundary such that local refinement on $\mathbf{x}$ is worth executing. This will be elaborated in greater detail in subsequent sections.

4.2 Feasibility Boundary

Some portion of the feasibility boundary, which discriminates the feasible from the infeasible candidate solutions, can possibly be approximated upon encountering a mixed neighborhood, which consists of both the feasible and the infeasible solutions. Let y^i be the feasibility class of the neighboring solution $\mathbf{x}^i$. The feasibility information can therefore be incorporated into the neighborhood $\mathcal{N}$ by redefining

$$\mathcal{N}(\mathbf{x}|k) = \{(\mathbf{x}^i, y^i) : \mathbf{x}^i \neq \mathbf{x}, \mathbf{x}^i \in \mathcal{DB}, rank(dist(\mathbf{x}, \mathbf{x}^i)) \leq k\} \tag{9}$$

where

$$y^i = \begin{cases} +1 & \forall j \ g_j(\mathbf{x}^i) \leq 0 \\ -1 & \exists j \ g_j(\mathbf{x}^i) > 0 \end{cases} \tag{10}$$

The existence of two and only two feasibility classes within the framework of constrained optimization recommends that a two-class classification problem can somehow be formulated. Classification problem of distinguishing the regions of feasible candidate solutions from those of the infeasible ones is most intuitive in the context of mixed neighborhood. Algorithm that handles classification problem generally produces a decision function $D(\mathbf{x})$ for n-dimensional candidate solution $\mathbf{x}$ and employs a zero-threshold between the two classes.

As a positive value has been assigned to the feasible candidate solutions while a negative one to the infeasible candidate solutions, it is thus desirable that any classification algorithm observes the following behaviors upon provision of k data instances $(\mathbf{x}^i, y^i)$ for $i = 1, 2, \ldots, k$ during the training phase.

- $D(\mathbf{x}^i) > 0$ if $y^i = +1$ ($\mathbf{x}^i$ is feasible)
- $D(\mathbf{x}^i) < 0$ if $y^i = -1$ ($\mathbf{x}^i$ is infeasible)

After some successful training, the decision function generated could then produce one of the following three outcomes given a candidate solution $\mathbf{x}$ that has never been seen before by the classification algorithm.

- $D(\mathbf{x}) > 0$, signifying that $\mathbf{x}$ is predicted to be feasible
- $D(\mathbf{x}) < 0$, signifying that $\mathbf{x}$ is predicted to be infeasible
- $D(\mathbf{x}) = 0$, defining the decision boundary

Viewing the decision boundary as the approximation of the feasibility boundary, the value of the decision function $D(\mathbf{x})$ may thus be used as an estimate of the distance measure between a candidate solution $\mathbf{x}$ and the feasibility boundary. In spite of possibilities that the decision boundary may accurately replicate the true feasibility boundary of the optimization problem within some locality, the following two cases of misclassification—as illustrated in Figure 2—may generally occur.

1. A feasible candidate solution $\mathbf{x}$ is predicted to be infeasible ($D(\mathbf{x}) < 0$)
2. An infeasible candidate solution $\mathbf{x}$ is predicted to be feasible ($D(\mathbf{x}) > 0$)

In both cases, it may be intuitive to claim the significance of candidate solution $\mathbf{x}$ for a local refinement recalling that the global optimum may be situated at some particular point along the feasibility boundary.

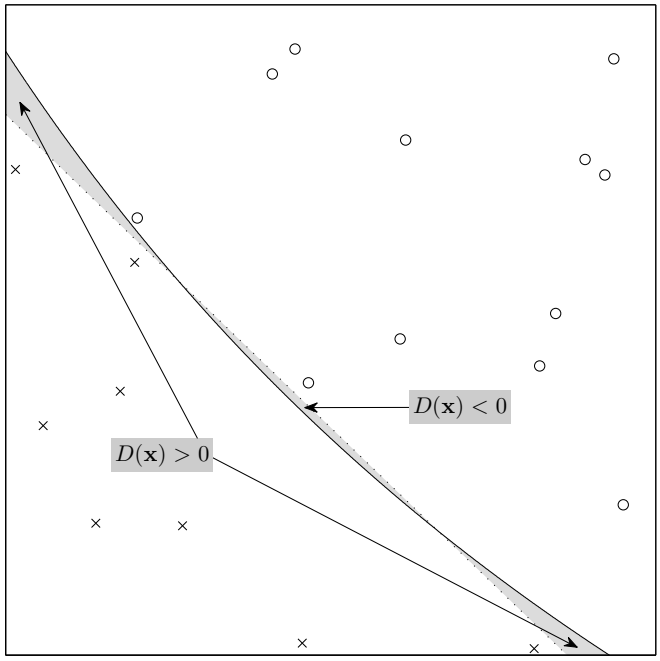

Figure 2: Phenomenon of Misclassification
The solid line is the true feasibility boundary while the dotted line the predicted one. Shaded areas represent the misclassified candidate solutions.

4.3 Support Vector Machine

Originally proposed for handling the linearly separable two-class classification problems [42], Support Vector Machine (SVM) [43, 44] minimizes classification error—while simultaneously maximizes the geometric margin of separation that sets the two classes apart. The SVM can be employed to solve the two-class classification problem established in the preceding section. On the provision of a set of k training data instances $\{(\mathbf{x}^i, y^i) : i = 1, 2, \ldots, k\}$, the SVM will aim at identifying an proper set of coefficients $\{\mathbf{w}, w_0\}$ that defines a linear decision function

$$D(\mathbf{x}) = (\mathbf{w} \cdot \mathbf{x}) + w_0 \tag{11}$$

that satisfies

$$\forall i \; y^i D(\mathbf{x}^i) \geq 1 \tag{12}$$

where $y^i \in \{+1, -1\}$. Kernel tricks [45] can be used for nonlinear decision functions [46].

The area where no training data instances may reside is termed the margin-of-separation. For a mixed neighborhood, such area signifies a region from which no feasibility information can actually be inferred. Exploitations are hence significant to reveal more about the region, especially if it has been known beforehand that the global optimum must have been situated along the feasibility boundary. Given by all $\mathbf{x}$ that satisfies $|D(\mathbf{x})| < 1$, it can be shown that the width of the margin-of-separation is $\frac{2}{\|\mathbf{w}\|}$; maximizing which is equivalent to minimizing

$$\frac{1}{2}\|\mathbf{w}\|^2 \tag{13}$$

subject to

$$\forall i \; y^i[(\mathbf{w} \cdot \mathbf{x}^i) + w_0] \geq 1 \tag{14}$$

in which the factor of $\frac{1}{2}$ is introduced only for mathematical convenience.

The corresponding dual to the above primal is the maximization of a quadratic program

$$\sum_{i=1}^{k} \alpha^i - \frac{1}{2} \sum_{i=1}^{k} \sum_{j=1}^{k} \alpha^i \alpha^j y^i y^j (\mathbf{x}^i \cdot \mathbf{x}^j) \tag{15}$$

subject to

$$\sum_{i=1}^{k} \alpha^i y^i = 0 \tag{16}$$

$$\forall i \; \alpha^i \geq 0 \tag{17}$$

Should a training data instance $\mathbf{x}^i$ have $\alpha^i > 0$, it is then a support vector. Each support vector occupies either decision surface ($D(\mathbf{x}) = +1$ or $D(\mathbf{x}) = -1$), depending on which class it belongs to. Having identified all support vectors, the weight vector $\mathbf{w}$ and the bias w_0 can then be computed using

$$\mathbf{w} = \sum_{i=1}^{k} \alpha^i y^i \mathbf{x}^i = \sum_{i \in SV} \alpha^i y^i \mathbf{x}^i \tag{18}$$

$$w_0 = \frac{1}{|SV|} \sum_{i \in SV} \left[y^i - \sum_{j=1}^{k} \alpha^j y^j (\mathbf{x}^i \cdot \mathbf{x}^j) \right] \tag{19}$$

in which SV denotes the set of indices of the support vectors.

4.4 Feasibility Structure Modeling

As neighborhood of a candidate solution hints the feasibility information of the surroundings and the SVM model constructed from a mixed neighborhood approximates the local segment of the feasibility boundary, hence, feasibility structure modeling on a locality can be achieved to assess the significance of a candidate solution for the local refinement. While it is possible to come upon highly nonlinear feasibility boundary, the nonlinearity will be captured only in the early generations of the search for the distribution of candidate solutions in the database should be fairly sparse. As the search progresses, however, linear approximation within some locality will suffice. As demonstrated in Figure 3 where the margin-of-separation of the SVM is shown as the shaded area, linear approximation to the feasibility boundary can be deemed adequate because the incorrectly predicted candidate solution would also be claimed as being close to some feasibility boundary. Altogether, both factors should augment the performance of the feasibility structure modeling without necessitating the use of some nonlinear decision boundary.

It is also important to be aware of the three scenarios that may be encountered following the evaluations of candidate solutions in the initial population.

1. All candidate solutions turn out to be feasible
2. Some candidate solutions are feasible and some are infeasible
3. All candidate solutions turn out to be infeasible

Whereas it should be intuitive to start to apply the relevant rules for the feasibility structure modeling in the first two scenarios, it may not sound so prudent to do so in the last scenario. As it can be very difficult for the global search to find feasible candidate solutions, especially in dealing with minute ratio of the feasible to the entire search space, local refinement needs to be executed on every candidate solution until a feasible candidate solution is found.

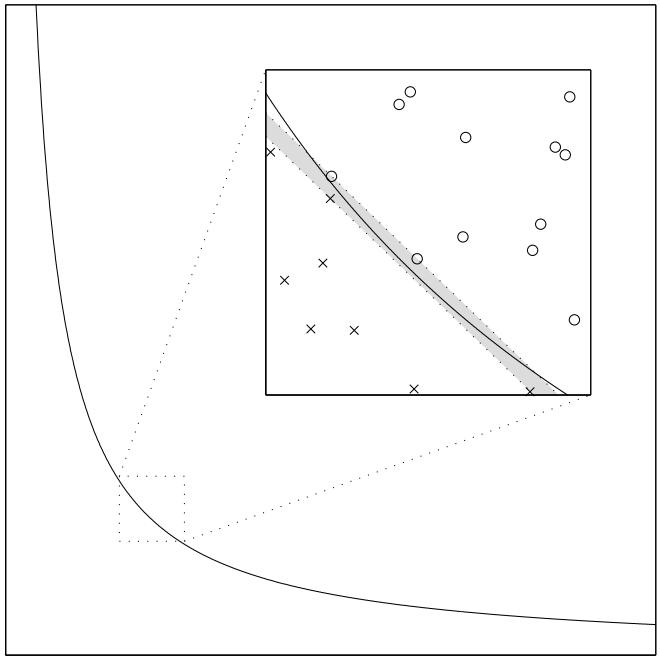

Figure 3: Linear Approximation of Local Feasibility Boundary

4.5 Adaptive Constrained Memetic Algorithms

In Algorithm 2, a general framework of the adaptive constrained MAs through modifications to the simple MA so as to enable assessment of the significance of each candidate solution in a population to undergo local refinement is presented. Being part of an attempt to minimize the overall requirements of the FFEs for identifications of the global optimum of constrained optimization problems, the assessment allows an adaptive selection of candidate solutions for local refinements. Simple GA is resembled should all the candidate solutions in a population be considered insignificant for local refinements. Simple MA is instantiated, on the contrary, if the candidate solutions in the population are considered significant for the refinements.

In this research work, three assessment strategies based on *Feasibility Structure Modeling* (FSM) are instantiated, hence the name FSM-based MAs. Intended for two different a priori knowledge of constrained optimization problems, Algorithm 3 and 4 present the FSM-ac and the FSM-ic. Algorithm 5 then instantiates the FSM for no a priori knowledge.

Algorithm 2 General Framework of Adaptive Constrained Memetic Algorithm

```
Initialize a population of candidate solutions
Evaluate the fitness of each candidate solution
for each candidate solution in the population do
  if the candidate solution is significant for local refinement then
    Refine the candidate solution locally
  end if
end for
while no stopping criteria have been fulfilled do
  Evolve the current population by one generation
  Evaluate the fitness of each candidate solution
  for each candidate solution in the population do
    if the candidate solution is significant for local refinement then
      Refine the candidate solution locally
    end if
  end for
end while
```

Detailed in Algorithm 3 is the assessment strategy for constrained optimization problems having assumed that the global optimum must have been located at some particular location along the feasibility boundary (FSM-ac). Most intuitively, a local refinement on a solution $\mathbf{x}$ should be executed upon encountering a mixed neighborhood. However, the local refinement must only be executed if $\mathbf{x}$ is predicted by the SVM to belong to the margin-of-separation—capturing the previously unexplored region—or is wrongly predicted by the SVM—signifying nonlinearity of the feasibility boundary the linear SVM fails to capture. As discussed before, a local refinement on $\mathbf{x}$ will also be necessary upon encountering the infeasible neighborhood provided that $\mathbf{x}$ is feasible, and vice versa. Supposedly, it is known a priori that the problem to solve is characterized by the global optimum along the feasibility boundary. It would then be advantageous to employ this assessment strategy in the context of the FSM-based MAs.

Algorithm 3 Assessment Strategy of FSM-based MA (Active Constraints)

function REFINE = ASSESS($\mathbf{x}$)
REFINE = FALSE
if the database $\mathcal{DB}$ has no feasible candidate solutions **then**
  REFINE = TRUE
else
  Extract the neighborhood structure $\mathcal{N}(\mathbf{x}|k)$
  if $\mathcal{N}(\mathbf{x}|k)$ is a mixed neighborhood **then**
    Obtain the decision function $D = \text{SVM}(\mathcal{N}(\mathbf{x}|k))$
    if $|D(\mathbf{x})| \leq 1$ OR $\mathbf{x}$ is misclassified **then**
      REFINE = TRUE
    end if
  else if $\mathcal{N}(\mathbf{x}|k)$ is a feasible neighborhood AND $\mathbf{x}$ is infeasible **then**
    REFINE = TRUE
  else if $\mathcal{N}(\mathbf{x}|k)$ is an infeasible neighborhood AND $\mathbf{x}$ is feasible **then**
    REFINE = TRUE
  end if
end if

Detailed in Algorithm 4 is the assessment strategy for constrained optimization problems having assumed that the global optimum must have been situated at the trough of a feasible basin (FSM-ic). Being unconstrained optimization alike—except that their search space need not necessarily be rectangular—it is therefore most intuitive to execute a local refinement on a feasible candidate solution with either a feasible or an infeasible neighborhood. An entirely unconstrained local search space is hinted by the earlier while the possibly disjointed feasible space is suggested by the latter. Upon an encounter with the mixed neighborhood, a feasible candidate solution with respect to which the neighborhood has been defined is significant for a local refinement if it is correctly predicted as feasible but not on the margin-of-separation, indicating a feasible search space that is sufficiently far from the feasibility boundary, or it is simply incorrectly predicted by the SVM, suggesting the possibly disjointed feasible space.

Algorithm 4 Assessment Strategy of FSM-based MA (Inactive Constraints)

function REFINE = ASSESS($\mathbf{x}$)
REFINE = FALSE
if the database $\mathcal{DB}$ has no feasible candidate solutions **then**
  REFINE = TRUE
else
  Extract the neighborhood structure $\mathcal{N}(\mathbf{x}|k)$
  if $\mathcal{N}(\mathbf{x}|k)$ is a mixed neighborhood **then**
    Obtain the decision function $D = \text{SVM}(\mathcal{N}(\mathbf{x}|k))$
    if $D(\mathbf{x}) \geq 1$ OR $D(\mathbf{x}) \leq 0$ AND $\mathbf{x}$ is feasible **then**
      REFINE = TRUE
    end if
  else
    if $\mathbf{x}$ is feasible **then**
      REFINE = TRUE
    end if
  end if
end if

Lastly, Algorithm 5 details the assessment strategy for constrained optimization problems making no assumptions whatsoever about the situation of the global optimum. The strategy may not be as efficient as the other two but it can be employed without necessarily requiring a priori knowledge on the problem to optimize. Focussing the search efforts on both possible locations of the global optimum, Algorithm 5 resembles Algorithm 3 except for two changes:

- Upon encountering a feasible neighborhood, local refinement is consistently carried out no matter if the candidate solution against which the neighborhood has been defined is feasible or infeasible.
- Upon encountering a mixed neighborhood, local refinement is performed additionally if the solution against which the neighborhood has been defined is predicted as feasible.

Algorithm 5 Assessment Strategy of FSM-based MA (General Case)

function REFINE = ASSESS($\mathbf{x}$)

REFINE = FALSE

if the database $\mathcal{DB}$ has no feasible candidate solutions **then**

REFINE = TRUE

else

Extract the neighborhood structure $\mathcal{N}(\mathbf{x}|k)$

if $\mathcal{N}(\mathbf{x}|k)$ is a mixed neighborhood **then**

Obtain the decision function $D = \text{SVM}(\mathcal{N}(\mathbf{x}|k))$

if $D(\mathbf{x}) \geq -1$ OR $\mathbf{x}$ is misclassified **then**

REFINE = TRUE

end if

else if $\mathcal{N}(\mathbf{x}|k)$ is a feasible neighborhood **then**

REFINE = TRUE

else if $\mathcal{N}(\mathbf{x}|k)$ is an infeasible neighborhood **then**

if $\mathbf{x}$ is feasible **then**

REFINE = TRUE

end if

end if

end if

4.6 Complexity Analysis

In this section, n_{eval} and n_{gen} is defined respectively as the number of FFEs and generations needed to optimize a problem using memetic algorithm that assumes a population size of m. Employing the proposed approach so as to minimize the FFE requirement, the following two components must then be performed in order to allow modeling of the feasibility structure of the problem.

1. Extracting the neighborhood structure $\mathcal{N}(\mathbf{x}|k)$

   It should be noted that the choice of the neighborhood size k must be reasonably small to ensure locality and, more importantly, to allow acceptable training time of the SVM but sufficiently large to reassure that information in the surroundings of the solution $\mathbf{x}$ is adequately represented. A neighborhood size of few multiple of the problem size, *i.e.* $k = \zeta n$ where ζ is a small positive integer, may be a reasonable choice. When $k \ll |\mathcal{DB}|$ where $|\mathcal{DB}|$ is the size of the database, a neighborhood may be identified using a k-pass algorithm where an "unmarked" past solution with minimum distance to $\mathbf{x}$ is identified and marked in each pass—yielding a time complexity of $\mathcal{O}(|\mathcal{DB}|)$. As the database size will be at its maximum at the end of the search with n_{eval} past solutions recorded, it is clear that time complexity of the neighborhood extraction is $\mathcal{O}(n_{\text{eval}})$ in the worst case.

2. Deducing the decision function $D \equiv \text{SVM}(\mathcal{N}(\mathbf{x}|k))$

   Upon an encounter with mixed neighborhood, the SVM needs training for deduction of the decision function D. Should all the k neighbors become the support vectors in SV, the time complexity of training the SVM is $\mathcal{O}(k^3)$ [47]. Assuming that $k = \zeta n$ where ζ is a reasonably small positive integer and n is the size of the problem, time complexity of the deduction of decision function D is therefore $\mathcal{O}(n^3)$.

The overall worst-case time complexity of solving a problem using the proposed approach is $\mathcal{O}(mn_{\text{gen}}(n_{\text{eval}} + n^3)) + \mathcal{O}_{\text{GS}} + \mathcal{O}_{\text{LS}} + \mathcal{O}_{\text{FFE}}$; the last three terms of which are respectively complexity of the entire global search, the entire local search, and all of the FFEs performed during the optimization process. In the worst case—although unlikely—local refinements are performed on each candidate solution in every generation. For such a case, $\mathcal{O}_{\text{LS}}$ is dependent on n_{eval}. Meanwhile, $\mathcal{O}_{\text{GS}}$ is always dependent on n_{gen} and $\mathcal{O}_{\text{FFE}}$ on n_{eval}. For it is common to observe $n_{\text{gen}} \ll n_{\text{eval}}$, $\mathcal{O}_{\text{GS}}$ is often negligible. Optimizing some computationally-expensive problems, $\mathcal{O}_{\text{FFE}}$ should then prevail dominating the other terms.

5 Results and Discussions

For the empirical study of the efficacy of the three instances of FSM-based MAs, experiments comprising 30 independent runs of the simple as well as the adaptive MAs were conducted on thirteen benchmark problems whose characteristics are tabularized in Table 1. Mathematical expressions of the objective and constraint functions of each problem can be found in [48].

Table 1: Characteristics of the Benchmark Problems

Problem	n	**Objective**	n_g	**Constraint(s)**	$\rho^{\dagger}$	$f(\mathbf{x}_{\text{best}})$
g01	13	quadratic	9	linear	0.0111%	−15.000
g02	20	nonlinear	2	polynomial	99.9971%	−0.803619
g04	5	quadratic	6	quadratic	52.1230%	−30665.539
g06	2	cubic	2	quadratic	0.0066%	−6961.814
g07	10	quadratic	8	quadratic	0.0003%	24.306
g08	2	nonlinear	2	quadratic	0.8560%	−0.095825
g09	7	polynomial	4	polynomial	0.5121%	680.630
g10	8	linear	6	quadratic	0.0010%	7049.248
g12	3	quadratic	1	quadratic	4.7713%	−1.000
g16	5	nonlinear	38	nonlinear	0.0204%	−1.905
g18	9	quadratic	13	quadratic	0.0000%	−0.866025
g19	15	cubic	5	quadratic	33.4761%	32.656
g24	2	linear	2	polynomial	79.6556%	−5.508

†: Ratio of the feasible to the entire search space [48]

During the course of experimentations, the following settings were consistently observed.

- *global search*: GA with a population size of 100 candidate solutions
- *local search*: SQP with a local refinement budget of
  - 10 FFEs for problems having *polynomial* objective functions (**g01**, **g04**, **g06**, **g07**, **g09**, **g10**, **g12**, **g16**, **g18**, **g19**, and **g24**)
  - 100 FFEs for problems involving *trigonometric* objective functions (**g02** and **g08**)
- *neighborhood size*: twice the dimensionality of the problem
- *terminating criterion*: global optimum identified with a precision of 10^{-10}

5.1 Memetic vs. Pure Evolutionary Algorithms

It will be shown in this section that memetic algorithm—represented by the simple MA (SMA) in which local refinement is conducted for every candidate solution in each generation—really outperforms the pure evolutionary algorithms in attempting to locate with sufficient precision the global optimum of the benchmark problems considered herein. The SMA is contrasted to three existing methods, namely the Stochastic Ranking (SR) [28], the Simple Multimembered Evolution Strategy (SMES) [27], and the so-called Triangular Crossover (TC) in tandem with a novel ranking scheme [31], in order to reveal that it is truly reasonable to make the SMA—which is hybridization of GA and SQP—reference for further comparisons in the next section on the first nine problems. The remaining four problems, however, were not addressed in these works. Hence, comparisons will be made to three other works from CEC 2006 that produced the best results [49, 50, 51]. The lack of results in solving constrained optimization problems using GA and SQP has deemed experimentations under such scheme with the most primitive strategy—the simple MA—important to ensure the fairness of further comparisons presented in the subsequent sections. As computational budget, SMES and SMA had used only 240,000 FFEs while SR and TC had used 350,000 FFEs. The other three algorithms from CEC 2006, however, had consumed up to 500,000 FFEs. It is then presented in Table 2 the average fitness of the solutions obtained upon exhaustion of the respective computational budget.

As can be seen from Table 2, the SMA had indeed performed significantly well compared to the pure evolutionary algorithms by solving completely all but problem **g02** despite using only 240,000 FFEs—the smallest computational budget among those of the existing methods. The average fitness of -0.803022 with standard deviation of 0.002272 from the solutions of problem **g02** attained by the SMA, which is better than those achieved by the three existing methods, further confirms that it is indeed reasonable to assess the newly proposed approach to realizing the adaptive constrained MAs with respect to the SMA in the following sections.

Table 2: Average Fitness of Solutions Obtained upon Exhaustion of Computational Budget

Problem	**SR** [28]	**SMES** [27]	**TC** [31]	**SMA**
g01	**−15.000**	**−15.000**	**−15.000**	**−15.000**
g02	−0.781975	−0.785238	−0.791345	−0.803022
g04	**−30665.539**	**−30665.539**	−30665.531	**−30665.539**
g06	−6875.940	−6961.284	**−6961.814**	**−6961.814**
g07	24.374	24.475	25.057	**24.306**
g08	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**
g09	680.656	680.643	680.659	**680.630**
g10	7559.192	7253.047	7493.719	**7049.248**
g12	**−1.000**	**−1.000**	**−1.000**	**−1.000**
Problem	[49]	[50]	[51]	**SMA**
g16	**−1.905**	**−1.905**	**−1.905**	**−1.905**
g18	**0.866025**	**0.866025**	**0.866025**	**0.866025**
g19	**32.656**	33.342	**32.656**	**32.656**
g24	**−5.508**	**−5.508**	**−5.508**	**−5.508**

Note: Number in boldface signifies the global optimum was reached in all runs

5.2 Simple vs. FSM-based Memetic Algorithms

Having shown in the preceding section that memetic algorithms (as represented by the SMA) are generally better than pure evolutionary algorithms, it will hence be shown in this section that FSM-based MAs, which are capable of determining intelligently if a candidate solution is significant for local refinement, enhance the efficacy of the simple MA. For efficiency of these different algorithms is the main focus of the study in this section, all runs of the algorithms were terminated only when the global optimum had been located with a precision of 10^{-10}. The three instances of the FSM-based MAs described previously had performed equally well if not better than the SMA. The scheme that tackles problems with global optimum at point along the feasibility boundary (FSM-ac) completely solved problem **g01**, **g02**, **g04**, **g06**, **g07**, **g09**, **g10**, **g16**, **g18**, **g19**, and **g24**. The scheme that handles problems with global optimum at trough of the feasible basin (FSM-ic) completely solved problem **g08** and **g12**. All of these problems were also completely solved by the general scheme that addresses problems without a priori knowledge about the situation of their global optimum (FSM).

5.2.1 Global Optimum at Point along Feasibility Boundary

Benchmark problem **g01**, **g02**, **g04**, **g06**, **g07**, **g09**, **g10**, **g16**, **g18**, **g19**, and **g24** are some inequality-constrained problems with global optimum located at some particular point along the feasibility boundary. In the context of FSM-based MAs, these problems may effectively be solved using either the FSM or the FSM-ac in which refinements are geared towards potential locations of the global optimum. Statistics on the amount of FFEs needed to completely solve the problems are tabulated in Table 3. These comprise the minimum, median, and maximum as well as the mean and standard deviation of the FFE requirements. Within one single row, lower value is more favorable and the best one in the row is presented in boldface for ease of comparison.

A quick glance at the table suggests that either one of the two schemes of the FSM-based MAs employed here had performed more efficiently than the SMA in solving all but problem **g01**. A thorough examination of the figures in the table then reveals that the minimum FFE requirements of the FSM-based MAs are consistently lower than those of the SMA across all problems. The maximum FFE requirements of the FSM-based MAs, however, are occasionally higher than those of the SMA with special emphasis on problem **g01**, **g16**, and **g19**. Except for problem **g01**, the average—and also, median—FFE requirements of the FSM-based MAs are lower than those of the SMA, raising a concern if the occasionally higher worst-case FFE requirements of the FSM-based MAs could have happened just by chance. In order to address the issue, statistical t-tests were conducted between two pairs of MAs: FSM-ac vs. SMA and FSM vs. SMA. The p-values from the tests reveal how likely it would be for the difference of the average FFE requirements to occur only by chance. The last possible pair of MAs—FSM vs. FSM-ac—was also assessed using the t-tests for significant difference in their performance. Both t-values and p-values as well as the winning algorithms under the 90%, 95%, and 99% confidence level are tabulated in Table 4.

Table 3: Statistics of the FFE Requirements of the Simple and FSM-based MAs for Problems with Global Optimum at Point along the Feasibility Boundary

Problem	Statistics	SMA	FSM-ac	FSM
g01	minimum	215	**103**	**103**
	median	**228**	294	294
	maximum	**237**	622	601
	mean ± st.dev.	**228 ± 5**	277 ± 139	281 ± 143
g02	minimum	63,728	**47,546**	57,163
	median	97,627	**70,857**	91,324
	maximum	466,986	**121,196**	307,084
	mean ± st.dev.	118,654 ± 75,959	**73,345 ± 17,928**	100,950 ± 48,766
g04	minimum	470	**197**	240
	median	478	**224**	269
	maximum	489	**264**	314
	mean ± st.dev.	479 ± **5**	**226** ± 18	274 ± 21
g06	minimum	1,047	**105**	**105**
	median	1,069	**110**	**110**
	maximum	1,079	**197**	**197**
	mean ± st.dev.	1,066 ± **8**	**116** ± 18	**116** ± 18
g07	minimum	4,380	**538**	**538**
	median	14,213	3,255	**2,225**
	maximum	62,286	29,334	**24,133**
	mean ± st.dev.	18,831 ± 15,243	6,009 ± 8,190	**3,952 ± 4,823**
g09	minimum	6,541	**994**	1,004
	median	18,559	6,881	**6,586**
	maximum	38,200	24,269	**15,021**
	mean ± st.dev.	18,964 ± 7,444	8,077 ± 5,494	**7,730 ± 4,419**
g10	minimum	1,976	**271**	**271**
	median	5,978	2,110	**1,712**
	maximum	19,676	9,535	**5,174**
	mean ± st.dev.	7,097 ± 4,637	2,975 ± 2,394	**2,093 ± 1,381**
g16	minimum	307	**287**	**287**
	median	377	**325**	**325**
	maximum	**540**	551	551
	mean ± st.dev.	396 ± 75	**366 ± 71**	**366 ± 71**
g18	minimum	2,176	**1,210**	**1,210**
	median	2,191	**1,875**	**1,875**
	maximum	7,647	**4,804**	5,635
	mean ± st.dev.	2,950 ± 1,255	**2,058 ± 998**	2,095 ± 1,075
g19	minimum	1,100	**912**	1,087
	median	2,189	**1,152**	1,370
	maximum	2,198	**1,638**	2,205
	mean ± st.dev.	2,043 ± 376	**1,160 ± 132**	1,441 ± 266
g24	minimum	508	**208**	321
	median	537	**258**	380
	maximum	572	**322**	444
	mean ± st.dev.	538 ± **18**	**257** ± 25	376 ± 30

Note: Number in boldface signifies the best one(s) in the row

Table 4: Statistical t-Test between Different Pairs of MAs for Problems with Global Optimum at Point along the Feasibility Boundary

Problem	**Algorithms**	t-**value**	p-**value**	**Winner**		
				90%	**95%**	**99%**
g01	FSM-ac vs. SMA	1.9053	6.1706×10^{-2}	SMA	–	–
	FSM vs. SMA	2.0056	4.9576×10^{-2}	SMA	SMA	–
	FSM vs. FSM-ac	0.1142	9.0946×10^{-1}	–	–	–
g02	FSM-ac vs. SMA	3.1798	2.3660×10^{-3}	FSM-ac	FSM-ac	FSM-ac
	FSM vs. SMA	1.0743	2.8715×10^{-1}	–	–	–
	FSM vs. FSM-ac	2.9101	5.1165×10^{-3}	FSM-ac	FSM-ac	FSM-ac
g04	FSM-ac vs. SMA	73.2831	7.1534×10^{-59}	FSM-ac	FSM-ac	FSM-ac
	FSM vs. SMA	53.0276	7.6524×10^{-51}	FSM	FSM	FSM
	FSM vs. FSM-ac	9.6527	1.1312×10^{-13}	FSM-ac	FSM-ac	FSM-ac
g06	FSM-ac vs. SMA	267.2613	2.4290×10^{-91}	FSM-ac	FSM-ac	FSM-ac
	FSM vs. SMA	267.2613	2.4290×10^{-91}	FSM	FSM	FSM
	FSM vs. FSM-ac	0.0000	1.0000	–	–	–
g07	FSM-ac vs. SMA	4.0585	1.4964×10^{-4}	FSM-ac	FSM-ac	FSM-ac
	FSM vs. SMA	5.0975	3.9426×10^{-6}	FSM	FSM	FSM
	FSM vs. FSM-ac	1.1857	2.4058×10^{-1}	–	–	–
g09	FSM-ac vs. SMA	6.4452	2.4885×10^{-8}	FSM-ac	FSM-ac	FSM-ac
	FSM vs. SMA	7.1081	1.9297×10^{-9}	FSM	FSM	FSM
	FSM vs. FSM-ac	0.2698	7.8830×10^{-1}	–	–	–
g10	FSM-ac vs. SMA	4.3259	6.0632×10^{-5}	FSM-ac	FSM-ac	FSM-ac
	FSM vs. SMA	5.6636	4.8534×10^{-7}	FSM	FSM	FSM
	FSM vs. FSM-ac	1.7472	8.5893×10^{-2}	FSM	–	–
g16	FSM-ac vs. SMA	1.6380	1.0683×10^{-1}	–	–	–
	FSM vs. SMA	1.6380	1.0683×10^{-1}	–	–	–
	FSM vs. FSM-ac	0.0000	1.0000	–	–	–
g18	FSM-ac vs. SMA	3.0498	3.4483×10^{-3}	FSM-ac	FSM-ac	FSM-ac
	FSM vs. SMA	2.8350	6.2975×10^{-3}	FSM	FSM	FSM
	FSM vs. FSM-ac	0.1394	8.8962×10^{-1}	–	–	–
g19	FSM-ac vs. SMA	12.1331	1.4901×10^{-17}	FSM-ac	FSM-ac	FSM-ac
	FSM vs. SMA	7.1650	1.5480×10^{-9}	FSM	FSM	FSM
	FSM vs. FSM-ac	5.1875	2.8385×10^{-6}	FSM-ac	FSM-ac	FSM-ac
g24	FSM-ac vs. SMA	49.8807	2.4676×10^{-49}	FSM-ac	FSM-ac	FSM-ac
	FSM vs. SMA	25.8474	1.6088×10^{-33}	FSM	FSM	FSM
	FSM vs. FSM-ac	16.6840	8.3213×10^{-24}	FSM-ac	FSM-ac	FSM-ac

Note: Dash (–) signifies the two algorithms are equally efficient

It can clearly be seen from the table that the SMA wins over both schemes of FSM-based MAs employed here only when solving problem **g01** with up to 95% confidence level. Solving the rest of the problems, however, both the FSM and the FSM-ac performed more efficiently than the SMA—except when solving problem **g02** and **g16** using the FSM and problem **g16** using the FSM-ac; for which they had performed as efficiently as the SMA.

Simplicity of the problem **g01**—a quadratic program having nine linear constraints—has made the SMA performed more efficiently than the FSM-based MAs on the average. In spite of greater best-case FFE requirements of the SMA, its significantly more stable performance has caused the SMA to win over both schemes of the FSM-based MAs. Local refinement for each candidate solution in the initial population escalates the chance of successfully locating the global optimum, which could have been located as soon as the end of the first generation for such simple problem like **g01**. Due to a relatively small ratio of the feasible to the entire search space, the initial population of candidate solutions to the problem **g01** must have been dominated by the infeasible ones. In the context of FSM-based MAs, a refinement on each of the candidate solutions in the population is only executed while no feasible solution has been found; after which the decision for the execution of a local refinement will depend entirely on the feasibility structure modeled. Should the refinement that found the first feasible solution enhance the "right" candidate solution, the global optimum would have been identified right away. This results in the more efficient best-case performance of the FSM-based MAs. There is no guarantee, nonetheless, that such phenomenon should consistently be observed in every run of the algorithms. Once the refinement that had found the first feasible solution actually converged to a local optimum, the FSM-based MAs then had to depend on the global search to find suitable candidate solution before applying the next refinement. Meanwhile, the SMA would just need to keep refining the subsequent candidate solution. With high probability of encountering the infeasible neighborhood, it is hence understandable that the convergence of the FSM-based MAs to the global optimum is possibly delayed whilst the global search looks for a suitable candidate solution with an indication that it is situated nearby some feasibility boundary. This vindicates the less efficient average-case and worst-case performance for both FSM-based MAs. Furthermore, the low probability of coming across a feasible neighborhood when solving problem **g01** has camouflaged the difference between the two FSM-based MAs, leading to minute difference in their performance.

In contrast to problem **g01**, problem **g02** holds an extremely large ratio of the feasible to the entire search space—resembling almost perfectly an unconstrained optimization problem. The feasible neighborhood is thus more easily encountered than the other two types. Should it be known that the global optimum is located at some point along the feasibility boundary, it will be efficient to avoid refinements of the candidate solutions with feasible neighborhood. The FSM-ac demonstrates such behavior, which apparently was more efficient than the SMA as suggested by the t-test in Table 4. In the absence of an a priori knowledge, refinements of the candidate solutions with feasible neighborhood should not be ignored. The FSM exhibits this behavior, which had apparently resulted in similar efficiency to the SMA as indicated by the t-test in Table 4. At best, the FSM refined most—but not all—of the candidate solutions in each generation. Therefore, similar efficiency to the SMA is indeed expected. Nonetheless, reduced FFE requirements are still observable from Table 3 when solving problem **g02** using the FSM. Although deemed insignificant by the t-test, such reductions are yet advantageous, especially when dealing with computationally-expensive problems where each evaluation may take several minutes to several hours to complete.

Sharing similarity to problem **g01** is problem **g16**—which is characterized by a relatively small ratio of the feasible to the entire search space. Hence, the initial population must have been dominated by infeasible candidate solutions. Being more complex nonlinear program—rather than simply a quadratic program—the problem has made the SQP unable to progress in many of its runs. Hence, simply performing a local refinement on every candidate solution in the initial population shall not dramatically escalate the chance of successful identification of the global optimum. This translates to an increased number of generations of the SMA in dealing with this problem. While it is only the FSM-based MAs that needed to depend much on the global search when solving problem **g01**, the SMA also demonstrated such a behavior in the optimization of problem **g16**. With the increase of the FFE requirements of the SMA, similar efficiency are observed among the different MAs employed here.

Except for the three special cases discussed above, both instances of the FSM-based MAs studied in this section are in general more efficient than the SMA. Statistical t-tests between the two FSM-based MAs further reveal that the FSM-ac had performed more efficiently than the FSM in handling problem **g02**, **g04**, **g19**, and **g24**; all of which are benchmark problems characterized by reasonably large ratio of the feasible to the entire search space. By avoiding local refinements on the feasible candidate solutions situated within feasible neighborhood or the "already-exploited" feasible space nearby feasibility boundary, the FSM-ac demonstrated that it is indeed able to cut the costs of function evaluations significantly. This is possible as encounters with feasible neighborhood are not uncommon during the course of the search for the global optimum. In handling problem **g01**, **g06**, **g07**, **g09**, **g10**, **g16**, and **g18**, however, it is found out that the FSM-ac had performed just as efficiently as the FSM. With the ratio of the feasible to the entire search space amounting to less than 1%, it is thus more common to encounter the infeasible rather than the feasible neighborhood. The intrinsic advantage of the FSM-ac therefore is somewhat concealed. Furthermore, although unlikely, there remains the possibility of encountering feasible neighborhood. As the relatively small ratio translates into minute feasible region(s), performing local refinements on the candidate solutions having feasible neighborhood will most likely end up in an optimum—potentially global optimum—though it might be located along the feasibility boundary. This proves beneficial for the FSM as it is affirmed to be more efficient than the FSM-ac with up to 90% confidence in handling problem **g10** as signified in Table 4. This also justifies the lesser amount of FFEs required by the FSM compared to that required by the FSM-ac that are consistently observed in Table 3 for the latter group of benchmark problems.

While we can have the privilege to know the global optimum of the benchmark problems and the access to sufficient computational resources to solve these problems until completion, it is often not the case when dealing with real-world problems. Table 5 shows the normalized fitness gap (20) averaged over 30 runs with the success rate under different amount of FFEs.

Table 5: Average Normalized Fitness Gap and Success Rate under Different Amount of FFEs for Problems with Global Optimum at Point along the Feasibility Boundary

Problem	#FFEs	SMA	FSM-ac	FSM
g01	500	0 (100%)	0.00012530 (93%)	0.00026913 (90%)
	1,000	0 (100%)	0 (100%)	0 (100%)
g02	500	0.73284574 (0%)	0.72871824 (0%)	0.73284574 (0%)
	1,000	0.72238410 (0%)	0.70502817 (0%)	0.72238410 (0%)
	5,000	0.65326714 (0%)	0.60868984 (0%)	0.65326714 (0%)
	10,000	0.60108992 (0%)	0.53093372 (0%)	0.59759099 (0%)
	50,000	0.28836665 (0%)	0.14892439 (3%)	0.23321994 (0%)
	100,000	0.01647077 (57%)	0.00082972 (93%)	0.00519557 (67%)
	500,000	0 (100%)	0 (100%)	0 (100%)
g04	500	0 (100%)	0 (100%)	0 (100%)
g06	500	0.00000000 (0%)	0 (100%)	0 (100%)
	1,000	0.00000000 (0%)	0 (100%)	0 (100%)
	5,000	0 (100%)	0 (100%)	0 (100%)
g07	500	0.06168829 (0%)	0.05426596 (0%)	0.05387976 (0%)
	1,000	0.05982416 (0%)	0.03686453 (13%)	0.03609042 (13%)
	5,000	0.00515322 (3%)	0.00065157 (70%)	0.00026621 (80%)
	10,000	0.00169745 (33%)	0.00030547 (87%)	0.00009164 (90%)
	50,000	0.00000993 (93%)	0 (100%)	0 (100%)
	100,000	0 (100%)	0 (100%)	0 (100%)
g09	500	0.44549137 (0%)	0.31167973 (0%)	0.31184970 (0%)
	1,000	0.27972736 (0%)	0.09210764 (3%)	0.08968011 (0%)
	5,000	0.00312079 (0%)	0.00102496 (27%)	0.00091745 (30%)
	10,000	0.00107018 (7%)	0.00008229 (77%)	0.00010987 (67%)
	50,000	0 (100%)	0 (100%)	0 (100%)
g10	500	0.03037484 (0%)	0.12783542 (7%)	0.12782474 (7%)
	1,000	0.02886063 (0%)	0.10327538 (23%)	0.09164043 (27%)
	5,000	0.00298767 (33%)	0.00054717 (80%)	0.00000002 (97%)
	10,000	0.00000729 (83%)	0 (100%)	0 (100%)
	50,000	0 (100%)	0 (100%)	0 (100%)
g16	500	0.00000000 (90%)	0.00000006 (97%)	0.00000006 (97%)
	1,000	0 (100%)	0 (100%)	0 (100%)
g18	500	0.00013736 (0%)	0.00013736 (0%)	0.00013736 (0%)
	1,000	0.00010018 (0%)	0.00010018 (0%)	0.00010018 (0%)
	5,000	0.00000415 (93%)	0 (100%)	0.00000150 (97%)
	10,000	0 (100%)	0 (100%)	0 (100%)
g19	500	0.00000000 (0%)	0.00000000 (0%)	0.00000000 (0%)
	1,000	0.00000000 (0%)	0.00000000 (10%)	0.00000000 (0%)
	5,000	0 (100%)	0 (100%)	0 (100%)
g24	500	0.00000000 (0%)	0 (100%)	0 (100%)
	1,000	0 (100%)	0 (100%)	0 (100%)

$$\text{Normalized Fitness Gap} = \frac{f(\mathbf{x}) - f(\mathbf{x}_{\text{best}})}{f(\mathbf{x}_{\text{worst}}) - f(\mathbf{x}_{\text{best}})} \tag{20}$$

Excluding problem **g01** as it can be solved efficiently by the SMA, the following is noted. Under very limited computational budget, it is observed that the average fitness gap values of the FSM-based MAs are either the same as or lower than those of the SMA for most problems except for **g10** and **g16**. This signifies that both instances of the FSM-based MAs are at least as close as the SMA to the global optimum of the benchmark problems. In fact, the FSM and the FSM-ac had been able to successfully completed more runs than the SMA while their gap values are not better than those of the SMA, particularly for problem **g10** and **g16** after 500 FFEs as well as **g10** and **g19** after 1,000 FFEs. Greater gap values for problem **g10** and **g16** can thus be construed to have come from the remaining uncompleted runs, which must have had worse fitness than most of the runs involving the SMA under the same amount of FFEs. The phenomenon can further be interpreted as the tendency of FSM-based MAs to hold back execution of local refinements while letting the global search explores the entire search space in order to generate potential solutions for the refinements. Slow convergence characteristics of the global search can ultimately be concluded to have caused the notably larger gap values under very limited computational budget. After some sufficient amount of FFEs, nonetheless, it is observed that the FSM-based MAs are reliably closer to the global optimum, resulting in faster convergence than the SMA.

As the FFE requirements of the two schemes of the FSM-based MAs are by and large lower than those of the SMA, savings of the CPU time taken to optimize computationally-expensive problems are anticipated. Reductions of tens to hundreds of FFEs translate easily into savings of hours to days of computational time should a single FFE takes minutes to hours to perform. Intriguingly, some accelerations were also observed when optimizing the benchmark problems considered herein, which are computationally inexpensive. Table 6 shows the total CPU time averaged over 30 runs, which was dedicated for the optimization of the benchmark problems. The table also presents separately the average time taken to search for the global optimum and the average time taken to decide which candidate solutions underwent local refinements.

Table 6: Average CPU Time Requirement (Seconds) of the Simple and FSM-based MAs for Problems with Global Optimum at Point along the Feasibility Boundary

Problem	Component	SMA	FSM-ac	FSM
g01	searching time	1.79	**0.32**	0.34
	modeling time	–	0.78	**0.77**
	total time	1.79	**1.10**	1.11
g02	searching time	2,463.70	**879.77**	1,186.23
	modeling time	–	**983.48**	1,085.96
	total time	2,463.70	**1,863.25**	2.272.19
g04	searching time	1.45	**0.50**	0.66
	modeling time	–	**0.31**	**0.31**
	total time	1.45	**0.81**	0.97
g06	searching time	1.79	**0.05**	**0.05**
	modeling time	–	**0.03**	**0.03**
	total time	1.79	**0.08**	**0.08**
g07	searching time	68.55	15.01	**9.55**
	modeling time	–	24.80	**11.02**
	total time	68.55	39.81	**20.57**
g09	searching time	37.84	12.57	**11.37**
	modeling time	–	13.69	**8.85**
	total time	37.84	26.26	**20.22**
g10	searching time	21.71	4.64	**3.10**
	modeling time	–	5.97	**3.34**
	total time	21.71	10.62	**6.44**
g16	searching time	14.26	**12.84**	**12.84**
	modeling time	–	**0.04**	**0.04**
	total time	14.26	**12.88**	**12.88**
g18	searching time	9.95	**6.71**	6.73
	modeling time	–	**0.60**	0.66
	total time	9.95	**7.31**	7.39
g19	searching time	10.87	**5.97**	7.28
	modeling time	–	**4.16**	4.46
	total time	10.87	**10.13**	11.74
g24	searching time	1.06	**0.40**	0.65
	modeling time	–	**0.08**	0.09
	total time	1.06	**0.48**	0.74

Note: Number in boldface signifies the best one(s) in the row

From the table, it can be observed that both FSM-based MAs had consistently entailed lower utilization of CPU time than the SMA—except for the FSM to optimize problem **g19**. While the CPU time requirements observed are in line with their FFE requirements for most problems, it is fascinating to observe how the negative reduction of FFEs between the SMA and the FSM-based MAs can lead to a positive saving of CPU time on problem **g01**.

Investigating the searching time, it is necessary to realize the following three contributing factors: the global search method for exploration, the local search technique for exploitation, and the fitness function evaluation of the problem being solved. For all benchmark problems considered herein, their fitness function evaluations are not computationally expensive, hence negligible. Furthermore, it is generally the case where the time demanded by the local search dominates the time needed to evolve the global search, hence $\mathcal{O}_{\text{LS}}$ searching time complexity. For the accurate analysis of the CPU time, it is important to note that the SQP employed in this empirical study requires solving direction-finding subproblems; each of which is of $\mathcal{O}(n^3)$ time complexity [52] with n being dimensionality of the problem. In the context of the SMA, executions of the SQP on all candidate solutions in each generation implies that a direction-finding subproblem is solved for each candidate solution generated in the course of searching for the global optimum. Complexity of the entire local refinements that have been performed during the search process is hence $\mathcal{O}_{\text{LS}} = n_{\text{eval}}\mathcal{O}(n^3)$. In the context of the FSM-based MAs, it is indeed arguable that $\mathcal{O}_{\text{LS}} < n_{\text{eval}}\mathcal{O}(n^3)$ for not all candidate solutions across generations are refined. Comparing the savings achieved on the searching time to the reductions attained on the FFEs, it is found out that higher percentage of time savings than the FFE reductions are consistent across all problems whenever the FSM-based MAs were employed—suggesting that really the dominant time consumption of local search is reduced from $\mathcal{O}_{\text{LS}} = n_{\text{eval}}\mathcal{O}(n^3)$ in the SMA to $\mathcal{O}_{\text{LS}} < n_{\text{eval}}\mathcal{O}(n^3)$ in the FSM-based MAs. Of specifically significant attention is the problem **g01**, for which negative FFE reductions had somehow caused positive savings on the searching time. With a relatively small ratio of the feasible to the entire search space, candidate solutions to this problem—many of which are infeasible—must have had infeasible neighborhood deeming them as not potential for local refinement. The FSM-based MAs then must have depended greatly on the evolution of the global search—hence, $\mathcal{O}_{\text{LS}} \ll n_{\text{eval}}\mathcal{O}(n^3)$. This rationalizes the positive savings witnessed on the searching time in spite of the negative FFE reductions of problem **g01**.

Observing closely the modeling time, it is noticed that the FSM-ac had taken longer time than the FSM to optimize problem **g07**, **g09**, and **g10**. Requiring more FFEs, this is indeed anticipated. It is also noticed that the FSM-ac had taken comparable time to the FSM when optimizing the other problems except **g02** and **g19** with differences that amount to less than one-tenth of a second. Demanding lesser FFEs to optimize problem **g01**, **g04**, **g18**, and **g24**, this phenomenon then suggests that the FSM-ac could have had longer evolutions—requiring more generations of the global search—than the FSM. With the modeling time complexity of $\mathcal{O}(mn_{\text{gen}}(n_{\text{eval}} + n^3))$ as given in Section 4.6, higher generation count translates to increased number of times the neighborhood identification and the feasibility boundary approximation should have been executed even though the latter depends on the number of times the mixed neighborhood were identified. However, higher generation count implies higher probability of encountering the mixed neighborhood. Additionally, it is judicious to reason that the smaller FFEs demanded translates to faster neighborhood identification due to smaller database size. As n_{eval} will eventually be larger than n^3 after certain threshold when n is small enough like the case of the benchmark problems considered herein, the FFE requirements that are above the threshold but produce sufficiently large difference will then result in significantly distinct modeling time. Shorter modeling time requested by the FSM-ac compared to the FSM when handling problem **g02** and **g19**, which is in agreement with their FFE requirements, signifies that the thousands of FFEs needed to solve them had caused n_{eval} to dominate n^3 such that adequate difference in the FFE requirements had yielded faster feasibility structure modeling for these problems. For problem **g18**, however, the difference of just 37 FFEs on the average had not curtailed the FSM-ac modeling time significantly though the few thousands of FFEs required to solve it might have caused n_{eval} to dominate n^3. Hence, it is understandable that the FSM-ac might have demanded shorter modeling time should problem **g01**, **g04**, and **g24** require a few thousands of FFEs to solve and sufficiently different FFE requirements between the FSM-ac and the FSM be observed in all these problems.

In summary, the FSM-based MAs were observed to have relied more on the global search than the SMA. This has resulted in the significantly lower searching time of both the FSM and the FSM-ac such that some savings may still be observed in the total running time of the algorithms even at the expense of the modeling time. This is how the negative reductions of FFEs observed for problem **g01** had arrived at positive savings of the CPU time. Exception, however, was observed when comparing the total CPU time taken by the SMA to the FSM in the optimization of problem **g19**. This was apparently caused by the insufficiently large difference between the FFE requirements of the two algorithms—signified by the comparable minimum and worse maximum FFE requirements of the FSM compared to the SMA. Although similarities were observed for problem **g16**, the ratio of the feasible to the entire search space of this problem is much lower than that of problem **g19** suggesting the mixed neighborhood is more easily encountered in **g19** than **g16**. This translates to higher modeling time complexity of problem **g19** requiring more savings on the searching time to be able to observe savings on the total time. This was achieved by the FSM-ac through significantly greater reduction of FFEs, but apparently not by the FSM. Upon optimizing computationally-expensive problems, however, such an observation will not be present as $\mathcal{O}_{\text{FFE}}$ will dominate both $\mathcal{O}_{\text{LS}}$ and the modeling time complexity, $\mathcal{O}(mn_{\text{gen}}(n_{\text{eval}} + n^3))$. The reduction of 602 FFEs achieved on the average performance of the FSM compared to the SMA would have translated easily to savings of CPU time.

5.2.2 Global Optimum at Trough of Feasible Basin

Characterized as inequality-constrained problems with global optimum at the trough of some feasible basin are the benchmark problem **g08** and **g12**, which can effectively be solved using either the FSM or the FSM-ic so as to direct local refinements towards potential locations of the global optimum. Statistics on the amount of FFEs required to tackle these two problems are summarized in Table 7 followed by the t-tests between pairs of MAs in Table 8.

Table 7: Statistics of the FFE Requirements of the Simple and FSM-based MAs for Problems with Global Optimum at Trough of the Feasible Basin

Problem	Statistics	SMA	FSM-ic	FSM
g08	minimum	3,959	**120**	136
	median	4,453	**436**	448
	maximum	5,124	**1,637**	2,411
	mean ± st.dev.	4,496 ± **289**	**519** ± 318	649 ± 535
g12	minimum	441	**102**	129
	median	492	**118**	176
	maximum	591	**162**	235
	mean ± st.dev.	501 ± 37	**122** ± **13**	178 ± 27

Note: Number in boldface signifies the best one(s) in the row

Table 8: Statistical t-Test between Different Pairs of MAs for Problems with Global Optimum at Trough of the Feasible Basin

Problem	Algorithms	t-value	p-value	Winner		
				90%	95%	99%
g08	FSM-ic vs. SMA	50.7178	9.6015×10^{-50}	FSM-ic	FSM-ic	FSM-ic
	FSM vs. SMA	34.6668	1.8156×10^{-40}	FSM	FSM	FSM
	FSM vs. FSM-ic	1.1431	2.5771×10^{-1}	–	–	–
g12	FSM-ic vs. SMA	52.4972	1.3550×10^{-50}	FSM-ic	FSM-ic	FSM-ic
	FSM vs. SMA	38.2056	8.1626×10^{-43}	FSM	FSM	FSM
	FSM vs. FSM-ic	10.2530	1.2195×10^{-14}	FSM-ic	FSM-ic	FSM-ic

Note: Dash (–) signifies both algorithms are equally efficient

From the tables, it can be seen that the FSM-based MAs employed here had consistently performed more efficiently than the SMA. With virtually zero p-values, the t-tests show that it is extremely unlikely for the differences on the FFE requirements to have occurred only by chance. Between the two FSM-based MAs, it is discovered that the FSM-ic had consistently required lesser FFEs than the FSM. The t-tests, however, only validate that the FSM-ic had performed more efficiently than the FSM when optimizing problem **g12**. While this problem has similar complexity to the problem **g01** experimented with in the previous section, except for a quadratic constraint, its reasonably large ratio of the feasible to the entire search space had allowed the FSM-based MAs to perform more efficiently than the SMA. Since it is easier to come across the feasible or mixed neighborhood in **g12** than **g01**, the algorithms need not depend too heavily on the global search alone—leading to more efficient performance.

For the sake of completeness, Table 9 depicts the search traces of the different algorithms by exhibiting snapshots of the normalized fitness gap (20) averaged over 30 runs—along with the success rates—under different amount of FFEs. While the search traces for problem **g12** are clear, hence need no further explanation, the similar behavior of problem **g08** to problem **g10** and **g16** again suggests the tendency of the FSM-based MAs to hold back executions of the local refinements until the potential candidate solutions had been found—bringing about faster convergence to the global optimum.

Table 9: Average Normalized Fitness Gap and Success Rate under Different Amount of FFEs for Problems with Global Optimum at Trough of the Feasible Basin

Problem	#FFEs	SMA	FSM-ac	FSM
g08	500	0.00184565 (0%)	0.00448602 (60%)	0.00499246 (53%)
	1,000	0.00000000 (0%)	0.00003203 (93%)	0.00098376 (80%)
	5,000	0.00000000 (97%)	0 (100%)	0 (100%)
	10,000	0 (100%)	0 (100%)	0 (100%)
g12	500	0.00000000 (57%)	0 (100%)	0 (100%)
	1,000	0 (100%)	0 (100%)	0 (100%)

Table 10 then tabulates the different components of CPU time averaged over the 30 runs of the different algorithms for the optimization of all benchmark problems considered herein. As expected, it is found out that the SMA had demanded longer time than both FSM-based MAs and the FSM had needed longer searching but equivalent modeling time to the FSM-ic; all of which are in agreement with the FFE requirements.

Table 10: Average CPU Time Requirement (Seconds) of the Simple and FSM-based MAs for Problems with Global Optimum at Trough of the Feasible Basin

Problem	Component	SMA	FSM-ic	FSM
g08	searching time	6.18	**0.32**	0.48
	modeling time	–	0.08	**0.07**
	total time	6.18	**0.40**	0.55
g12	searching time	0.73	**0.05**	0.15
	modeling time	–	0.10	**0.09**
	total time	0.73	**0.15**	0.24

Note: Number in boldface signifies the best one(s) in the row

6 Real-World Problem: Potential Energy Minimization

After observing better performance of the FSM-based MAs on thirteen benchmark problems, it is then time to apply the proposed method to real-world problem: molecular conformation determination by means of potential energy minimization [53]. The molecule of the attention is the polyalanine (polypeptide made of alanine) that is eminent for its helical structure [54]. Recognition of the conformations of molecules is important as lots of reactions—especially in biochemistry [55]—obey the lock-and-key mechanism. Knowing the molecular conformations of the constituents of some reaction, hence, enables us to predict whether or not the reaction will take place. This is beneficial in the field of drug designs where there are needs to predict the behaviors of the drugs prior to clinical trials.

Figure 4 describes how amino acids are linked to each other in order to form polypeptide. The covalent bond formed between a carbon atom of one amino acid and a nitrogen atom of another amino acid allows long chains of amino acids to be constructed. Along the backbone of any single chain, the ϕ, ψ, and ω torsion angles describe the rotations about N–C^{α}, C^{α}–C, and C–N bonds, respectively. Thus, the torsion angle governs the distance between any atom before and any atom after a particular bond. Fine-tuning the torsion angles, therefore, varies the potential energy of the molecule as shall be seen later from the potential energy function. In addition to the backbone torsions, there are also torsion angles that describe the rotations about bonds on the side chains (*cf.* residue R in Figure 4). These are typically denoted as χ_i where the meaningful index i fluctuates from one amino acid to another. For an alanine with a methyl group ($-CH_3$) residue, there is only χ_1 governing the relative position of the group with respect to the remaining atoms in the polypeptide. While all ϕ, ψ, and χ torsion angles can take any value between -180° and 180°, subject to some steric constraints, the ω torsion angle is always around 180° [55]. The steric constraints prevent non-bonded atom pairs from overlapping each other, imposing restrictions on the feasible values of the torsion angles [56].

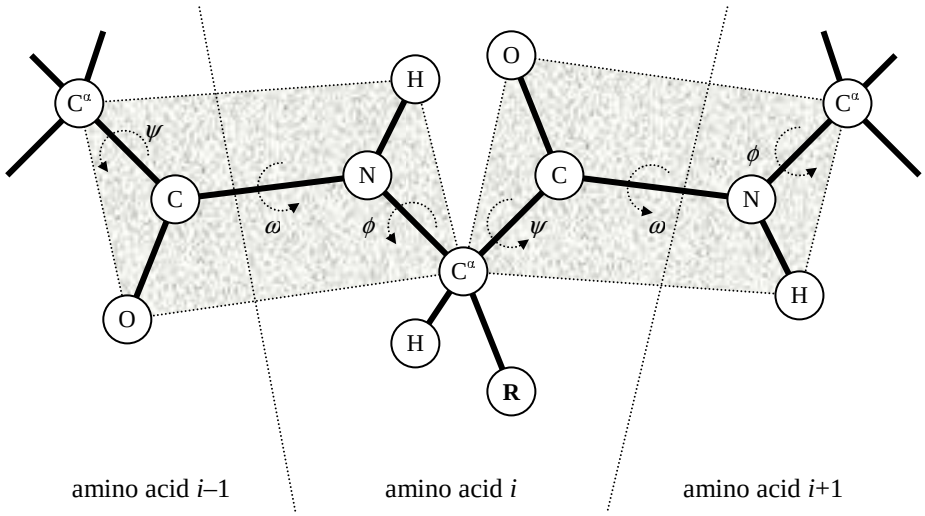

Figure 4: Polypeptide Chain

As the objective function is the following potential energy function employing parameters adopted from [57].

$$E = \sum_{\varphi} k_{\varphi} \left[1 + \cos\left(n\varphi - \gamma\right)\right] + \sum_{i<j} \left[\frac{q_i q_j}{r_{ij}} + \left(\frac{A_{ij}}{r_{ij}^{12}} - \frac{B_{ij}}{r_{ij}^{6}}\right)\right] \tag{21}$$

The torsional term is summed over all torsion angles φ whereas the non-bonded ones over all possible atom pairs (i, j) connected by three bonds or above. Given a set of torsion angles φ, the positions of all atoms can be deduced from which the interatomic distances r_{ij} may then be calculated. Fine-tuning φ changes r_{ij}, which altogether varies the potential energy E.

As the constraint function, it has been translated into a mathematical expression the fact that none of the non-bonded atom pairs may overlap each other. Treating each atom as hard sphere, the distance between any atom pair must then be larger than the sum of both atoms' radius. Mathematically,

$$R_i + R_j - r_{ij} < 0 \tag{22}$$

where R_i and R_j are the van der Waals radius [58, 59] of atom i and atom j, respectively, in which atom pairs (i, j) are connected by three or more bonds.

In the optimization of N-alanine, where N denotes the number of alanine's that compose the polypeptide, the design variables are the $3N$ torsion angles: ϕ, ψ, and χ_1 in each alanine. In the experiment, $N = 10$—accounting for 30 design variables and $4,968$ constraints—while the maximum number of FFEs is set to $40,000$. The experimental result plotted in Figure 5 suggests that both FSM-based MAs, FSM and FSM-ic, had performed better than the SMA. The FSM eliminated local refinement in the infeasible search space—preventing unnecessary function evaluations that must have been requested by the SMA. The FSM-ic further refined the choice of candidate solutions that should undergo local refinements by ruling local search out of the regions nearby some feasibility boundary. This scheme is intuitive for this problem as the minimum energy conformation of a molecule has r_{ij} strictly greater than $R_i + R_j$ such that the global optimum solution should reside at the trough of a feasible basin.

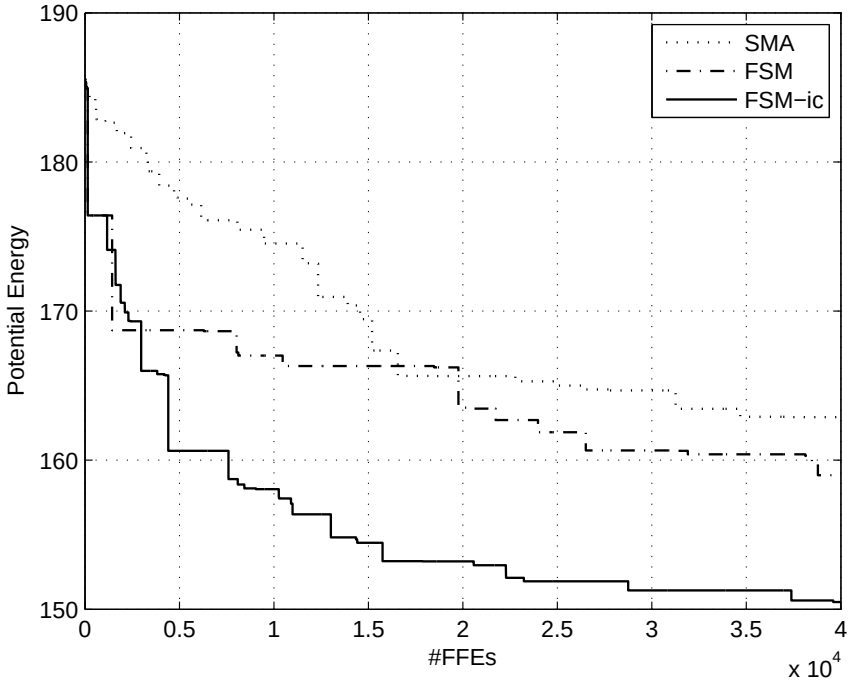

Figure 5: Trace of the Minimum Potential Energy while Optimizing Polyalanine

7 Conclusion

This paper, in the entirety, has been dedicated to address the problem of selecting promising candidate solutions to experience local refinement in the context of constrained optimization. It is first introduced in Section 4 the novel concept of modeling the feasibility structure given an inequality-constrained optimization problem that involves extraction of the neighborhood structure, approximation of some feasibility boundary, and prediction of the relative location of candidate solutions in the n-dimensional solution space, *i.e.* whether the solutions seem to be situated deep inside some feasible region, nearby some feasibility boundary, or deep inside some infeasible region. The proposed concept was then instantiated into the three schemes of the FSM-based MAs elaborated in Section 4.5. In Section 5, the efficacy of these algorithmic instances was investigated with a conclusion that the FSM-based MAs in general outperform the simple MA in optimizing the benchmark problems considered in this study. It is fortified further by the result attained when applying the relevant MAs to solve a real-world problem: minimization of the potential energy of a polyalanine elaborated in Section 6. As an integral part of the feasibility structure modeling, new paradigm of using classification in the context of constrained MAs is also introduced in this paper. A classifier—instead of a regressor as in the surrogate modeling [39, 40, 41]—is incorporated into the framework of MAs.

Hence, the main contribution of the works presented in this paper is the novel concept of utilizing information gathered along the evolution of MA to model the feasibility structure of constrained optimization problems and of incorporating classification—rather than regression for fitness approximations—into the framework of MAs. The proposition of the three schemes of FSM-based MAs as well as the use of Support Vector Machine as the classifier is only one among other possibilities. This has therefore opened wide the opportunities for future works, which may include studies on the database-related aspects of the proposed approach.

References

[1] M.N. Le, Y.S. Ong, and Q.H. Nguyen, "Optinformatics for Schema Analysis of Binary Genetic Algorithms," in *Proceedings of the 2008 Genetic and Evolutionary Computation Conference*, pp. 1121–1122, 2008.

[2] W.E. Hart, "Adaptive Global Optimization with Local Search," Ph.D. Thesis, University of California, 1994.

[3] M.H. Lim, S. Gustafson, N. Krasnogor, and Y.S. Ong, "Editorial to the First Issue," *Memetic Computing*, 1(1), pp. 1–2, 2009.

[4] B. Zhao, K.R. Sakharkar, C.S. Lim, P. Kangueane, and M.K. Sakharkar, "MHC-Peptide Binding Prediction for Epitope-based Vaccine Design," *International Journal of Integrative Biology*, 1(2), pp. 127–140, 2007.

[5] P.P. Bonissone, R. Subbu, N. Eklund, and T.R. Kiehl, "Evolutionary Algorithms + Domain Knowledge = Real-world Evolutionary Computation," *IEEE Transactions on Evolutionary Computation*, 10(3), pp. 256–280, 2006.

[6] M. Levitt, M. Hirshberg, R. Sharon, and V. Daggett, "Potential Energy Function and Parameters for Simulations of the Molecular Dynamics of Proteins and Nucleic Acids in Solution," *Computer Physics Communications*, 91, pp. 215–231, 1995.

[7] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance Between Genetic Search and Local Search in Memetic Algorithms for Multiobjective Permutation Flowshop Scheduling," *IEEE Transactions on Evolutionary Computation*, 7(2), pp. 204–223, 2003.

[8] M.W.S. Land, "Evolutionary Algorithms with Local Search for Combinatorial Optimization," Ph.D. Thesis, University of California, 1998.

[9] N.K. Bambha, S.S. Bhattacharyya, J. Teich, and E. Zitzler, "Systematic Integration of Parameterized Local Search into Evolutionary Algorithms," *IEEE Transactions on Evolutionary Computation*, 8(2), pp. 137–155, 2004.

[10] Q.H. Nguyen, Y.S. Ong, and M.H. Lim, "A Probabilistic Memetic Framework," *IEEE Transactions on Evolutionary Computation*, 13(3), pp. 604–623, 2009.

[11] D.H. Wolpert and W.G. MacReady, "No Free Lunch Theorems for Optimization," *IEEE Transactions on Evolutionary Computation*, 1(1), pp. 67–82, 1997.

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

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

[14] K. Schittkowski, "NLPQL: A FORTRAN Subroutine Solving Constrained Nonlinear Programming Problems," *Annals of Operations Research*, 5(2), pp. 485–500, 1986.

[15] E. Spedicato, *Algorithms for Continuous Optimization: The State of the Art*, Springer, 1994.

[16] W. Karush, "Minima of Functions of Several Variables with Inequalities as Side Conditions," M.Sc. Dissertation, Department of Mathematics, University of Chicago, 1939.

[17] H.W. Kuhn and A.W. Tucker, "Nonlinear Programming," in *Proceedings of the 2nd Berkeley Symposium on Mathematical Statistics and Probability*, pp. 481–492, 1950.

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

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

[20] D.E. Goldberg, *Genetic Algorithms for Search, Optimization, and Machine Learning*, Addison-Wesley, 1989.

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

[22] C.A.C. Coello, "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.

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

[24] H.J.C. Barbosa and A.C.C. Lemonge, "A New Adaptive Penalty Scheme for Genetic Algorithms," *Information Science*, 156(3–4), pp. 215–251, 2003.

[25] R. Farmani and J.A. Wright, "Self-Adaptive Fitness Formulation for Constrained Optimization," *IEEE Transactions on Evolutionary Computation*, 7(5), pp. 445–455, 2003.

[26] P. Chootinan and A. Chen, "Constraint Handling in Genetic Algorithms Using a Gradient-based Repair Method," *Computers and Operation Research*, 33(8), pp. 2263–2281, 2006.

[27] E. Mezura-Montes 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.

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

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

[30] P.Y. Ho and K. Shimizu, "Simple Addition of Ranking Method for Constrained Optimization in Evolutionary Algorithms," in *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation*, pp.889–896, 2005.

[31] E.Z. Elfeky, 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.

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

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

[34] N. Krasnogor and J. Smith, "A Memetic Algorithm with Self-adaptive Local Search: TSP as a Case Study," in *Proceeding of the 2000 Genetic and Evolutionary Computation Conference*, pp. 987–994, 2000.

[35] P. Merz, "Advanced Fitness Landscape Analysis and the Performance of Memetic Algorithms," *Evolutionary Computation*, 12(3), pp. 303–325, 2004.

[36] Y.S. Ong and A.J. Keane, "Meta-Lamarckian Learning in Memetic Algorithms," *IEEE Transactions on Evolutionary Computation*, 8(2), pp. 99–110, 2004.

[37] Y.S. Ong, M.H. Lim, N. Zhu, and K.W. Wong, "Classification of Adaptive Memetic Algorithms: A Comparative Study," *IEEE Transactions on Systems, Men, and Cybernetics – Part B*, 36(1), pp. 141–152, 2006.

[38] S. Hasan, R. Sarker, D. Essam, and D. Cornforth, "Memetic Algorithms for Solving Job-shop Scheduling Problems," *Memetic Computing*, 1(1), pp. 69–83, 2009.

[39] Y.S. Ong, P.B. Nair, and A.J. Keane, "Evolutionary Optimization of Computationally-expensive Problems via Surrogate Modeling," *American Institute of Aeronautics and Astronautics*, 41(4), pp. 687–696, 2003.

[40] Z.Z. Zhou, Y.S. Ong, M.H. Lim, and B.S. Lee, "Memetic Algorithm using Multi-Surrogates for Computationally-expensive Optimization Problems," *Soft Computing: A Fusion of Foundations, Methodologies, and Applications*, 11(10), pp. 957–971, 2007.

[41] Z.Z. Zhou, Y.S. Ong, P.B. Nair, A.J. Keane, and K.Y. Lum, "Combining Global and Local Surrogate Models to Accelerate Evolutionary Optimization," *IEEE Transactions On Systems, Man, and Cybernetics - Part C*, 37(1), pp. 66–76, 2007.

[42] V.N. Vapnik and A. Lerner, "Pattern Recognition Using Generalized Portrait Method," *Automation and Remote Control*, 24, 1963.

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

[44] B. Schölkopf, C.J.C. Burges, and S. Mika, *Advances in Kernel Methods*, The MIT Press, 1998.

[45] M. Aizerman, E. Braverman, and L. Rozonoer, "Theoretical Foundations of the Potential Function Method in Pattern Recognition Learning," *Automation and Remote Control*, 25, pp. 821–837, 1964.

[46] B.E. Boser, I.M. Guyon, and V.N. Vapnik. "A Training Algorithm for Optimal Margin Classifiers," in *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, pp. 144–152, 1992.

[47] C.J.C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," *Data Mining and Knowledge Discovery*, 2, pp. 121–167, 1998.

[48] J.J. Liang, T.P. Runarsson, E. Mezura-Montes, M. Clerc, P.N. Suganthan, C.A.C. Coello, and K. Deb, *Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization*, Technical Report, 2006.

[49] T. Takahama and S. Sakai, "Constrained Optimization by the ε Constrained Differential Evolution with Gradient-based Mutation and Feasible Elites," in *Proceedings of the 2006 Congress on Evolutionary Computation*, pp. 1–8, 2006.

[50] E. Mezura-Montes, J. Velázquez-Reyes, and C.A.C. Coello, "Modified Differential Evolution for Constrained Optimization," in *Proceedings of the 2006 Congress on Evolutionary Computation*, pp. 25–32, 2006.

[51] A. Sinha, A. Srinivasan, and K. Deb, "A Population-based, Parent Centric Procedure for Constrained Real-parameter Optimization," in *Proceedings of the 2006 Congress on Evolutionary Computation*, pp. 239–245, 2006.

[52] D. Goldfarb and S. Liu, "An $\mathcal{O}(n^3L)$ Primal Interior Point Algorithm for Convex Quadratic Programming," *Mathematical Programming*, 49, pp. 325-340, 1991.

[53] D.J. Wales, *Energy Landscapes*, Cambridge University Press, 2003.

[54] D. Poland and H.A. Scheraga, *Theory of Helix-Coil Transitions in Biopolymers*, Academic Press, 1970.

[55] D. Voet and J.G. Voet, *Biochemistry*, Wiley & Sons, 2004.

[56] G.N. Ramachandran, C. Ramakrishnan, and V. Sasisekharan, "Stereochemistry of Polypeptide Chain Configurations," *Journal of Molecular Biology*, 7, pp. 95–99, 1963.

[57] Y. Duan, C. Wu, S. Chowdhury, M.C. Lee, G. Xiong, W. Zhang, R. Yang, P. Cieplak, R. Luo, T. Lee, J. Caldwell, J. Wang, and P. Kollman, "A Point-charge Force Field for Molecular Mechanics Simulations of Proteins Based on Condensed-phase Quantum

Mechanical Calculations," *Journal of Computational Chemistry*, 24(16), pp. 1999–2012, 2003.

[58] A. Bondi, "van der Waals Volumes and Radii," *Journal of Physical Chemistry*, 68(3), pp. 441-451, 1964.

[59] R.S. Rowland and R. Taylor, "Intermolecular Nonbonded Contact Distances in Organic Crystal Structures: Comparison with Distances Expected from van der Waals Radii," *Journal of Physical Chemistry*, 100(18), pp. 7384-7391, 1996.