Hybrid Evolutionary Algorithm for Solving General Variational Inequality Problems*

Majig Mend-Amar, Abdel-Rahman Hedar and Masao Fukushima
Department of Applied Mathematics and Physics,
Graduate School of Informatics, Kyoto University, Kyoto 606-8501, Japan

August 2006

Abstract

This paper considers the problem of finding as many as possible, hopefully all, solutions of the general (i.e., not necessarily monotone) variational inequality problem (VIP). Based on global optimization reformulation of VIP, we propose a hybrid evolutionary algorithm that incorporates local search in promising regions. In order to prevent searching process from returning to the already detected global or local solutions, we employ the tunneling and hump-tunneling function techniques. The proposed algorithm is tested on a set of test problems in the MCPLIB library and numerical results indicate that it works well in practice.

Key words: Variational inequality, global optimization, evolutionary algorithm, local search, tunneling function.

1 Introduction

Let X be a nonempty closed convex set in R^n and $F : R^n \to R^n$ be a continuously differentiable mapping. The *variational inequality problem* (VIP) is to find a vector $x^* \in X$ such that

$$F(x^*)^T(x - x^*) \geq 0, \quad \forall x \in X. \tag{1.1}$$

This problem is denoted VIP(X, F) and has a large number of important applications. We refer the interested reader to the two volume book by Facchinei and Pang [3].

Theoretical aspects of the VIP have been studied well and many algorithms, such as projection methods, interior and smoothing methods and equation reduction methods, have been proposed to solve it [3, 4]. A popular idea adopted by recent algorithms is to reformulate the VIP as a system of equations or an optimization problem. However, the validity and efficiency of those algorithms often depend on the monotonicity-like assumption on the mapping F.

Although some algorithms have been proposed for solving general (not necessarily monotone) VIPs [1, 8, 9, 10, 14, 15, 17], they are primarily designed to solve the particular VIP(X, F) in which the constraint set X is the non-negative orthant R^n_+. This special problem, VIP(R^n_+, F), is called the *nonlinear complementarity problem*, and is denoted NCP(F). Moreover, most of the existing algorithms aim at finding a solution of this problem. In practice, however, it is desirable to find all, or as many as possible, solutions of the problem.

In this paper, we propose a global optimization-based method for finding as many as possible, hopefully all, solutions of the general VIP. To achieve this, we first use an optimization

*This research was supported in part by a Grant-in-Aid for Scientific Research from Japan Society for the Promotion of Science.

reformulation of VIP (1.1) based either on a merit function or on its KKT system. In either case, the VIP (1.1) is reformulated as the following box constrained optimization problem with *zero global minimum value* [3, 4]:

$$\min f(x) \quad \text{s.t. } x \in D, \tag{1.2}$$

where f is a real-valued function and the set D is defined as $D = \{x \in R^n |\ l \leq x \leq u\}$. Here $l, u \in R^n \cup \{\pm\infty\}$ are, possibly infinite, lower and upper bounds on the variable.

In order to find all global solutions of problem (1.2), we propose a population-based *hybrid evolutionary algorithm* (HEA) that incorporates local search in promising regions. The proposed method tries to keep and improve diversity of good trial points in the population set while searching for global minimizers of the objective function. Moreover, every time a global or local solution, or an unpromising trial point is detected by the HEA, the objective function of the problem is locally modified around this point to prevent the searching process from returning back to the vicinity of this solution again. Actually, the proposed HEA invokes some known strategies of hybrid metaheuristics [11, 16, 18] with some modifications to fit the general VIP.

Tunneling function method for finding a global minimum of a non-convex function was first introduced in [13]. Main idea of this method is that every time a local solution is detected in computation, it tries to construct a new objective function which has the same global minima as the original function but the detected local minimum is no longer a local minimum for the new function. The new function is treated as the objective function in the next search stage, and this process is repeated until a global solution is found. Another idea to escape from a detected local minimum, called the filled function method, is proposed in [5, 19]. Instead of constructing a tunnel at a detected local minimum, it considers a new objective function which has a local maximum at the detected local solution and has no stationary point at local solutions worse (having greater original objective function value) than the detected one. Both tunneling and filled function techniques are applied to the general nonlinear complementarity problem in [8], where a semi-smooth Newton method is presented.

In our method, the tunneling function technique is used not only for escaping from the detected local minimum, but also and more importantly for escaping from a detected global minimum and its basin to search for other global minimizers. However, the direct use of the tunneling function technique at a detected *global* minimum can be effective only if it is an exact solution of the problem. In practice, we can only expect to find approximations of global minima. To cope with this difficulty, before using the tunneling modification at a detected approximate global minimum, we suggest first to use another modification of the objective function, which is called a *hump* function and helps capture the exact global minimum near the detected approximate solution, and then construct a *hump-tunneling* function to which the HEA is applied.

The global optimal value of problem (1.2) is known to be zero. The proposed method HEA exploits this fact in two ways. First, it helps the HEA to determine whether a solution is global or not. Second and more importantly, if a modified objective function (either by a tunneling or by a hump-tunneling function) has at least one common global minimum with the original objective function, it must also have the zero global minimum value, i.e., the global minimum value of the objective functions will remain the same during the computation except when there are no other common global minimizers.

The organization of this paper is as follows: In Section 2, we first give a review of merit functions and global optimization reformulations of VIP. In Section 3, we describe the HEA and its elements in detail. The basic ideas behind the tunneling and hump function techniques

are also contained in this section. We then present numerical results in Section 4 and conclude the paper in Section 5.

2 Global Optimization Formulations of VIP

Consider the VIP, which is to find a vector $x^* \in D$ such that

$$F(x^*)^T(x - x^*) \geq 0, \quad \forall x \in D.$$

Definition. A merit function for the VIP is a nonnegative function $\theta : D \to R_+$ such that x^* is a solution of the VIP if and only if $x^* \in D$ and $\theta(x^*) = 0$. That is, the solutions of the VIP coincide with the global optimal solutions of the problem

$$\min \theta(x) \ \text{ s.t. } x \in D, \tag{2.1}$$

whose optimal objective value is zero.

There are some well known merit functions for VIP [3, 4].
1. Gap function:

$$\theta_{gap}(x) = \max_{y \in D} F(x)^T(x - y).$$

2. Natural residual function:

$$\theta_{NR}(x) = \|x - \Pi_D(x - F(x))\|,$$

where $\Pi_D(z) := \operatorname*{argmin}_{y \in D} \|z - y\|$ denotes the projection of point z on D.
3. Regularized gap function:

$$\begin{aligned} \theta_{reg}(x) &= \max_{y \in D}(F(x)^T(x - y) - \frac{1}{2}\|x - y\|^2) \\ &= F(x)^T(x - \Pi_D(x - F(x))) - \frac{1}{2}\|x - \Pi_D(x - F(x))\|^2. \end{aligned} \tag{2.2}$$

The first two merit functions are generally non-differentiable, while the third one is a continuously differentiable function.

Another way to define VIP as a global optimization problem with zero global minimum value is to use its KKT system. Consider the following VIP: Find $x^* \in \tilde{D}$ such that

$$F(x^*)^T(x - x^*) \geq 0, \quad \forall x \in \tilde{D} := \{x \in R^n |\ h(x) = 0,\ g(x) \leq 0\} \tag{2.3}$$

where $h : R^n \to R^{m_1}$ and $g : R^n \to R^{m_2}$ are an affine function and a continuously differentiable convex function, respectively. Although we can directly use one of merit function formulations of problem (2.3) as described earlier, computing them with the set $\tilde{D}$ would be rather expensive in general. The KKT system of problem (2.3) is given by

$$\begin{aligned} F(x) + \nabla h(x)\mu + \nabla g(x)\lambda &= 0 \\ h(x) &= 0 \\ g(x) \leq 0,\ \lambda \geq 0,\ \lambda^T g(x) &= 0, \end{aligned} \tag{2.4}$$

which is a mixed complementarity system. It has been shown [3] that, under some constraint qualification for $\tilde{D}$, the solutions of problem (2.3) coincide with the solutions of the KKT system (2.4). It is not difficult to see that the system (2.4) is equivalent to some box constrained VIP, so we can use a merit function formulation for that VIP and have a global optimization problem with zero global minimum value.

3 Hybrid Evolutionary Algorithm

A genetic or evolutionary algorithm applies the principles of evolutionary process observed in nature for finding a solution of a problem [2, 6]. An evolutionary algorithm for optimization is different from classical optimization methods in several aspects:
- It relies on random sampling. This makes it a nondeterministic method, for which there is no theoretical guarantee to find an optimal solution.
- While classical optimization methods maintain a single best solution found so far, an evolutionary algorithm maintains a *population* of candidate solutions. Only a few of these are best, but the other members of the population set are trial points in other regions of the search space, where a better solution may later be found. The use of population sets helps the evolutionary algorithm avoid being trapped at a local optimum.
- Inspired by the role of reproduction and mutation processes in the evolution of living things, an evolutionary algorithm tries to combine and change elements of existing solutions in order to create a new solution, with some of the features of parents. The elements of existing solutions are combined in a *crossover* operation. Moreover, random changes or *mutations* are made periodically for some members of the current population, thereby yielding a new candidate solution. There are many possible ways to perform crossover and mutation operations [7].
- An evolutionary algorithm performs a selection process in which the most fit members of the population survive, and the least fit members are eliminated. This process guides the population in an evolutionary algorithm towards ever-better solutions.

A drawback of any evolutionary algorithm is that a solution is judged better only in comparison to currently known other solutions; such an algorithm actually has no reasonable way to test whether a solution is, even local, optimal. This drawback will disappear when the minimum objective value is known, and the global optimization problem considered in this paper precisely meets this requirement.

Now we describe our hybrid evolutionary algorithm HEA for the following optimization problem with *zero* global minimum value:

$$\min f(x) \quad \text{s.t. } x \in D, \tag{3.1}$$

where the constraint set is defined as $D := \{x \in R^n |\ l \le x \le u\}$ with $l, u \in R^n \cup \{\pm\infty\}$. Assume that f is continuously differentiable. Our purpose is to design an evolutionary algorithm which is able to find as many solutions as possible of problem (3.1).

If x^* is a solution (global or local) of problem (3.1), then it must satisfy the following optimality condition:

$$\nabla f(x^*)^T (x - x^*) \ge 0, \ \forall x \in D. \tag{3.2}$$

We can rewrite the condition (3.2) as

$$g_i(x) := \frac{\partial f(x)}{\partial x_i} \begin{cases} \ge 0 & \text{if } x_i = l_i; \\ = 0 & \text{if } l_i < x_i^* < u_i \ ; \quad i = 1, \ldots, n. \\ \le 0 & \text{if } x_i = u_i, \end{cases}$$

Let us define the index sets

$$A_1^\varepsilon(x) = \{i |\ l_i \le x_i \le l_i + \varepsilon\}, \ A_2^\varepsilon(x) = \{i |\ l_i + \varepsilon < x_i < u_i + \varepsilon\}, \ A_3^\varepsilon(x) = \{i |\ u_i - \varepsilon \le x_i \le u_i\}$$

and the function

$$err^\varepsilon(x) = \sum_{i \in A_1^\varepsilon(x)} |\min\{g_i(x), 0\}| + \sum_{i \in A_2^\varepsilon(x)} |g_i(x)| + \sum_{i \in A_3^\varepsilon(x)} \max\{g_i(x), 0\},$$

which is defined on the set D. It is not difficult to see that if x^* is a solution (global or local) of problem (3.1), then for sufficiently small $\varepsilon > 0$ the value of the function $err^{\varepsilon}(x)$ at the point x^* is zero, i.e., $err^{\varepsilon}(x^*) = 0$.

In order to search for many global solutions simultaneously, the proposed evolutionary algorithm first tries to keep diversity in the population set. Due to the rules of accepting a newly produced trial solution to survive in the population set, most evolutionary algorithms have the tendency that population sets eventually cluster around only a few solutions. This is because, in ordinary evolutionary algorithms, a new trial solution is usually accepted to survive and replace some solution in the population set, if it is better than that. Although some algorithms such as scatter search method [11, 12] try to keep diversity, the number of different good points in the population set is still small (even if the objective function has many global solutions) and the remaining points are usually just diversity points. The HEA uses the Population Update Rules (see Section 3.1), which are novel types of criteria for accepting new trial solutions to survive in the population set, and tries to keep diversity while searching for promising points.

The HEA collects the detected global or local solutions, or unpromising trial points in the set S of modification points. Once one of those points is detected, the HEA adds it to S and modifies the objective function around this point in order to avoid returning to it again in further search. We employ tunneling or hump-tunneling function modification (see Section 3.2) with detected solutions to construct new objective functions, which have the same minimum points as the original objective function except those solutions already detected. Moreover, to achieve faster convergence, we apply a local optimization method starting from the best points in the population set. Although we use modified objective functions in the evolutionary search, we always use the original objective function in the local search.

To terminate the HEA, we use the following three different criteria.

-The number of function evaluations exceeds the pre-defined limit.

-The number of detected global solutions exceeds the pre-defined number.

-Let N_s be a pre-specified positive integer. If the most recently added N_s elements of the set S of modification points were not new global solutions, then we terminate the main algorithm.

The main loop of the proposed algorithm is stated as follows. Explanation of its components will be given later in detail.

HEA Algorithm

1. Initialization. Choose a population size M and fix parameters $m, l_s, N_s, N_k, \beta, \varepsilon_1, \varepsilon_2, \varepsilon_3 > 0$.
Initialize the set of modification points as $S := \emptyset$ and set the current objective function

$$f_c(x) := f(x).$$

Use the Diversification Generation Method to construct an initial population set P. Evaluate the trial points in P and order them according to their current objective function values so that x^1 is the best solution and x^M is the worst, i.e.,

$$f_c(x^1) \leq f_c(x^2) \leq \cdots \leq f_c(x^M).$$

Set the generation counter $t := 1$.

2. Parents Pool Generation. Generate a parents pool P', which consists of all different

pairs of the population set P.

3. Crossover and Mutation. Select a pair (p^1, p^2) from P'. Apply the Crossover and Mutation Procedure to the pair (p^1, p^2) to obtain two new solutions c^1, c^2.

4. Population Update. Using the Population Update Rule with c^1 and c^2, update the population set P. Delete the pair (p^1, p^2) from the parents pool P'. If $P' = \emptyset$, then go to Step 5; otherwise go to Step 3.

5. Modification of Objective Function. If for some $\bar{x} \in P$,

$$f(\bar{x}) \leq \varepsilon_1 \text{ (global solution), or}$$

$$err^{\varepsilon_3}(\bar{x}) \leq \varepsilon_2 \text{ (stationary point),}$$

then add $\bar{x}$ to the set S of modification points. Applying the procedure Modification of the Objective Function to the current objective function $f_c(x)$ with the point $\bar{x}$, reconstruct the objective function $f_c(x)$. Use the Diversity Generation Method to produce M new trial points and add them to the population set P. Reorder the elements in the set P according to their new objective function values and redefine the population set P as the best M elements in it. Continue with the new objective function $f_c(x)$ and population set P.

6. Stopping Condition. If one of the stopping conditions holds, then terminate the algorithm and refine the global solutions in S by some local search method. Otherwise, set $t := t + 1$ and go to Step 7.

7. Intensification. If the best solution in the population set P has not been improved enough, i.e., the objective value has not been decreased by a pre-specified fraction $\beta \in (0, 1)$ in the last N_k steps, then apply the local search method with l_s steps to the original objective function $f(x)$ starting from the best m elements of the set P. Delete the points used as starting points in local search from the set P and add m points produced by Diversification Generation Method to the set P. After the local search steps are taken, compare the values of the current objective function $f_c(x)$ at each pair of the starting point and the newly found point.

If the value at the newly found point is greater, then go to Step 7.1. Otherwise, go to Step 7.2.

7.1. We regard the starting point as an unpromising trial point and add it to the set S of modification points. Applying the procedure Modification of the Objective Function to the current objective function $f_c(x)$ with the starting point, reconstruct the objective function $f_c(x)$ and reorder the elements in the population set P according to their new objective function values. Go to Step 2.

7.2. Using the Population Update Rule, update the population set P with the newly found points. Go to Step 2.

Now we elaborate the steps used in the HEA.

Diversification Generation Method. The purpose of the diversification generation [11] is to generate a well distributed set of trial solutions. The basic Diversification Generation method uses controlled randomization and frequency memory to generate a set of diverse solutions. This can be accomplished by dividing the range $[l_i, u_i]$ of each variable into 4 sub-ranges of equal size. Then, a solution is constructed in two steps. First a subrange is randomly selected. The probability of selecting a subrange is determined to be inversely proportional to its frequency count. Then a value is randomly generated within the selected subrange.

Crossover and Mutation Procedure. The purpose of crossover is to produce children who are expected to possess better properties than their parents. Good results can be obtained with a random matching of the individuals [2, 6]. Some well known crossovers are the following [7].

Single-point crossover: One crossover position (coordinate) in the vector of variables (genes) is randomly selected and the variables situated after this point are exchanged between individuals, thus producing two offsprings.

Multi-point crossover: Some crossover positions are chosen, and then the variables between successive crossover points are exchanged among the two parents to produce new offsprings.

Intermediate recombination: The values of the offspring variables are chosen from the values of the parents variables according to some rule.

Although we could use various types of existing crossovers, we propose another crossover which may hopefully be more appropriate to our problem. We use the well known fact [3] that

$$x \text{ solves the VIP}(F, D) \iff x = \Pi_D(x - F(x)),$$

where $\Pi_D(z) := \underset{y \in D}{\operatorname{argmin}} \|z - y\|$ is the projection of point z on D. If we denote the mapping $H(x) := \Pi_D(x - F(x))$, then we have

$$\|x - \Pi_D(x - F(x))\| = \|x - H(x)\| = \sqrt{\sum_{i=1}^{n} (x_i - H_i(x))^2}.$$

Here $H_i(x)$ is i-th component of the vector $H(x)$. According to this formula, we may tell to some extent the quality of the gene x_i, that is, the smaller the value $|x_i - H_i(x)|$, the better the gene. In particular, the equalities $x_i - H_i(x) = 0,\ i = 1, \ldots, n$, hold at any solution of the VIP. Taking into account these properties, we propose the following crossover and mutation.

Let (p^1, p^2) be a pair of solutions used to produce new trial solutions.

Crossover. Let $\bar{p}$ denote the vector whose coordinates are given by

$$\bar{p}_j := \begin{cases} p_j^1, & \text{if } |p_j^1 - H_j(p^1)| \le |p_j^2 - H_j(p^2)|, \\ p_j^2, & \text{otherwise,} \end{cases} \qquad j = 1, \ldots, n.$$

Choose random numbers r_1, r_2 from the interval $[0, 1]$. Define two new trial solutions as follows: If $\bar{p} \neq p^1$ or $\bar{p} \neq p^2$, then

$$c^i := p^i + r_i(\bar{p} - p^i),\ i = 1, 2.$$

Otherwise,

$$c^1 := p^1 + r_1(p^2 - p^1), \quad c^2 := \begin{cases} \Pi_D(p^1 - r_2(p^2 - p^1)), & \text{if } \bar{p} = p^1; \\ \Pi_D(p^2 - r_2(p^1 - p^2)), & \text{if } \bar{p} = p^2. \end{cases}$$

Mutation. Choose random numbers r_1, r_2 from the interval $[0, 1]$. Define two new trial solutions as follows:

$$c^i := p^i + r_i(H(p^i) - p^i),\ i = 1, 2.$$

In the HEA, we use the above Crossover and Mutation in addition to the multi-point crossover to generate the children.

3.1 Population Update Rule

As we mentioned earlier, most evolutionary algorithms have the property that their population sets tend to cluster around only a few global solutions. Here we propose two different techniques to update the population set, which are aimed to keep diversity while searching for global solutions. The first one is somewhat heuristic and depends on the structure of the population

set. The second one is based on some tolerance parameter for the distance between trial points.

Population Update 1. Consider a set of points $X = \{x^1, x^2, \ldots, x^M\}$ sorted according to their objective function values. Let x be a trial solution used to update the population set.
1. If $f(x) \geq f(x^M)$, i.e., x is worse than the worst element in X, then discard x.
2. If $f(x) \leq f(x^1)$, i.e., x is better than the best element in X, then add x to X and delete the closest point to x in X.
3. If $f(x^i) \leq f(x) < f(x^{i+1})$, then let

$$k := \operatorname*{argmin}_{1 \leq j \leq i} \|x - x^j\|, \quad l := \operatorname*{argmin}_{i+1 \leq j \leq M} \|x - x^j\|.$$

Namely, x^k is the closest point to x among such points in X that their objective function values are smaller than $f(x)$, while x^l is the closest point to x among such points in X that their objective function values are greater than $f(x)$.
If $\|x - x^k\| \leq \|x^k - x^l\|$, then discard x.
If $\|x - x^k\| > \|x^k - x^l\|$ and $\|x - x^l\| \leq \|x^k - x^l\|$, then delete x^l from X and add x to X in the $(i+1)$-th position.
Otherwise, delete x^M from X and add x to X in the $(i+1)$-th position.

Population Update 2. Let $X = \{x^1, x^2, \ldots, x^M\}$ be a set of points sorted according to their function values, and $\varepsilon_D > 0$ be a fixed tolerance for the distance. Let x be a trial solution. Define

$$B(x, \varepsilon) := \{y \in R^n | \ \|x - y\| < \varepsilon\}, \quad k(i) := \operatorname*{argmin}_{1 \leq j \leq i} \|x - x^j\|.$$

1. If $f(x) \leq f(x^1)$, then add x to the set X and delete from X all the points x^j satisfying $x^j \in B(x, \varepsilon_D)$. If there is no such element in X, then delete x^M from X. If there are many, add new trial solutions generated by using Diversification Generation Method to X to keep the size of the population set P equal to M.
2. If $f(x^i) < f(x) \leq f(x^{i+1})$, then do the following:
If $x \in B(x^{k(i)}, \varepsilon_D)$, then discard x. Otherwise, add the point x to X, and delete all the elements x^j, $j = i+1, \ldots, M$ of X satisfying $x^j \in B(x, \varepsilon_D)$. If there is no such element in X, then delete x^M from X. If there are many, add new trial solutions generated by using Diversification Generation Method to X to keep the size of the population set P equal to M.

If $\varepsilon_D = 0$, then the Population Update Rule 2 will coincide with the ordinary update rule used in the genetic algorithm that accepts a child to survive if it is better than an element in the population.

3.2 Modification of the Objective Function

Let $f_c(x)$ be the current objective function used in the HEA and $\bar{x}$ be a point around which the function $f_c(x)$ is to be modified. Depending on the type of point $\bar{x}$, we use two different modifications. Recall that we can recognize whether $\bar{x}$ is a global solution or not, since we know that the global minimum value is zero.

Suppose first that $\bar{x}$ is not a global solution. According to the HEA, $\bar{x}$ must then be either a local solution or an unpromising trial point which lies in the basin of some detected solution. To avoid inefficient search around it in the next search stage, we use a new objective function which is constructed from the current objective function by augmenting the function value

around $\bar{x}$.

Tunneling function. Consider the following function:

$$f_t(x, \bar{x}) := f_c(x) \cdot \exp\Big(\frac{1}{\|x - \bar{x}\|^2}\Big). \tag{3.3}$$

This function is called a *tunneling function* because of its behavior around the point $\bar{x}$. For the sake of computational convenience, instead of directly using the function $f_t(x, \bar{x})$, we use the following approximation of this function:

$$\bar{f}_t(x, \bar{x}) := f_c(x) \cdot \exp\Big(\frac{1}{\varepsilon_t + \frac{1}{\rho_t^2}\|x - \bar{x}\|^2}\Big), \tag{3.4}$$

where ε_t and ρ_t are positive parameters that control the degree and the range of modification. Since $\bar{x}$ is not a global minimum and the objective function value is zero at any global minimum, the modified function $\bar{f}_t(x, \bar{x})$ has the same global minima as the function $f_c(x)$ has.

Now let $\bar{x}$ be an isolated global minimum of $f_c(x)$. Our purpose is to construct a new objective function which has the same global minimizers as the objective function $f_c(x)$ has, except $\bar{x}$. Moreover, we require the new function to have no solution around $\bar{x}$. In principle, we may use the tunneling function (3.3). If $\bar{x}$ is an exact global solution, i.e., $f_c(\bar{x}) = 0$, then under mild condition the function $f_t(x, \bar{x})$ can satisfy our requirements. But, if $\bar{x}$ is just an approximation of a global solution $\bar{x}^*$, as one may expect in practice, then it may not be appropriate to use the tunneling function modification $f_t(x, \bar{x})$, because the exact solution $\bar{x}^*$ still satisfies $f_t(\bar{x}^*, \bar{x}) = 0$ unexpectedly (see Figure 1).

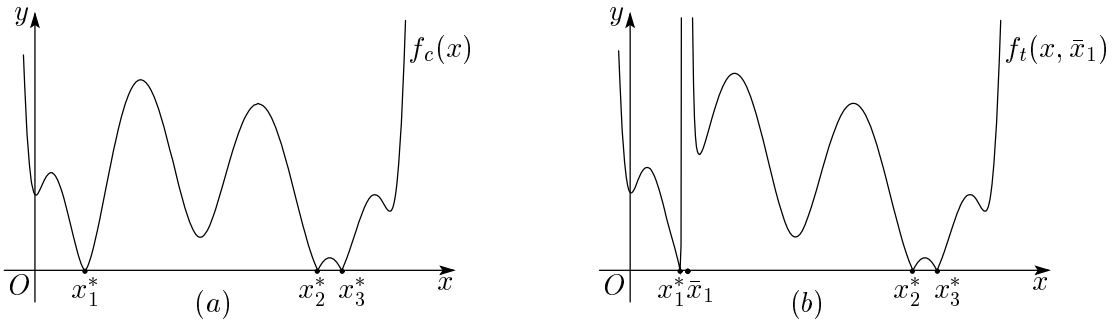

Figure 1: Graphs of a function (a) and its tunneling modification at an approximate solution $\bar{x}_1$ (b).

Below we propose a possible remedy to overcome the above-mentioned drawback of the tunneling function method.

Hump-Tunneling function. Consider first the following modified function:

$$f_h(x, \bar{x}) := f_c(x) + \alpha_h \max\Big\{0, 1 - \frac{1}{\rho_h^2}\|x - \bar{x}\|^2\Big\}, \tag{3.5}$$

where $\alpha_h, \rho_h > 0$ are some parameters. We call this function the *hump function* and this function may enable us to escape from the region around $\bar{x}$ even when $\bar{x}$ is an approximation of a global solution. However, it is not clear how we can determine the parameter ρ_h appropriately. If we set it smaller than necessary, the modified function may not be very useful because of a

narrow range of modification, and if we set it large, it may affect some other global solutions near $\bar{x}$, if any, and may make them non-global solutions any more (see Figure 2.a). Although it is not very appropriate to use either tunneling or hump function method individually, it may be effective to use a combination of these two functions.

We first take a sufficiently small positive scalar $\bar{\rho}_h$ and define a hump function $f_h(x, \bar{x})$ as in (3.5). Then we construct the following function:

$$\begin{aligned} \bar{f}_{ht}(x, \bar{x}) &:= f_h(x, \bar{x}) \cdot \exp\Big(\frac{1}{\varepsilon_t + \frac{1}{\rho_t^2}\|x - \bar{x}\|^2}\Big) \\ &= \Big(f_c(x) + \alpha_h \max\Big\{0, 1 - \frac{1}{\bar{\rho}_h^2}\|x - \bar{x}\|^2\Big\}\Big) \cdot \exp\Big(\frac{1}{\varepsilon_t + \frac{1}{\rho_t^2}\|x - \bar{x}\|^2}\Big). \end{aligned} \tag{3.6}$$

We call this function the *hump-tunneling function* and zero points of this function coincide with those of the function $f_c(x)$ except for those zeros in $B(\bar{x}, \bar{\rho}_h)$. Choosing $\bar{\rho}_h$ small enough, we can avoid affecting other global solutions near $\bar{x}$ (see Figure 2.b).

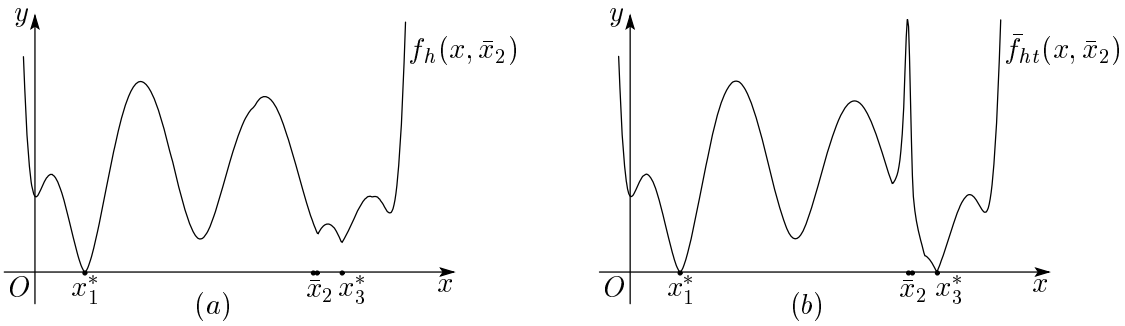

Figure 2: Graphs of a hump function (a) and a hump-tunneling function (b) constructed through modification at an approximate solution $\bar{x}_2$ of the function of Figure 1.a.

4 Numerical Experiments

The performance of the HEA was tested on a number of well known test problems in the MCPLIB library. To show the efficiency of the HEA we have used only those problems which have multiple solutions, and for each problem we made 20 trials with different initial populations. The programming code for the algorithm was written in MATLAB and run on a computer Pentium 4, Microprocessor.

The merit function (2.2) is used to reformulate MCPLIB test problems as optimization problems, and for local search in the HEA, we employ MATLAB's command `fmincon` combined with an active set detecting strategy. Moreover, the HEA is supposed to use a finite box for generating diversity points in the Diversification Generation Method, whereas most of the test problems are mixed complementarity problems which have no lower or upper bound. To deal with such problems, we use a fixed finite box defined inside the original box in the Diversification Generation Method, while using the original box constraint in the Crossover Mutation Procedure and in the local search.

In general, it is difficult to universally determine suitable values of HEA parameters for every problem, because they are highly problem dependent. Nevertheless, through testing

Table 1: Parameter Settings.

Parameters	Definition	Value
M	number of elements in population	$\min\{2n+4, 20\}$
m	number of best points for which local search is used	2
l_s	maximum number of steps per local search	$\min\{2n, 30\}$
N_k, β	parameters controlling local search in HEA	3, 0.999
$\varepsilon_1, \varepsilon_2, \varepsilon_3$	tolerance parameters for the objective function in HEA	$10^{-6}, 10^{-6}, 10^{-3}$
N_{max}	maximum number of ineffective local transformations	10
N_{gmax}	maximum number of global solutions to be found	20
NF_{max}	maximum number of function evaluations	$5n10^4$
ε_D	distance tolerance used in Population Update Rule 2	$n/5$
ε_t, ρ_t	tunneling parameters used in (3.4) and (3.6)	0.1, 2
α_h, ρ_h	humping parameters used in (3.6)	1, 0.3

many times on various test problems, we suggest possible choices of the parameters as shown in Table 1.

We have two versions of the HEA; HEA_1 and HEA_2 that use Population Update Rule 1 and Rule 2, respectively. We ran the HEA versions for all the chosen test problems with the general parameter settings mentioned in Table 1 and put the numerical results in Tables 2 and 3. The columns in these tables have the following meanings:

Problem:	name of the test problem,
n:	dimension of the test problem,
K_{min}, K_{av}, K_{max}:	minimum, average, maximum numbers of solutions found by the algorithm,
N_{gen}:	average number of generations,
N_{loc}:	average number of local steps taken,
NF:	average number of function evaluations,
N_f:	average number of function evaluations when the last global solution is obtained.

The results reported in Tables 2 and 3 show that the HEA is promising. For most of test problems, the average numbers of obtained global solutions (K_{av}) are close to the maximum numbers of obtained global solutions (K_{max}), and this implies that the HEA versions are capable of finding multiple solutions. Moreover, the average numbers of generations are reasonable compared with the problem dimensions and the numbers of obtained global solutions. The HEA versions use three different stopping conditions. Specifically, if the number of global solutions or that of ineffective transformations (i.e., the number of subsequently detected local solutions or unpromising trial points) or that of function evaluations exceeds their respective pre-specified limits $N_{gmax}, N_{max}, NF_{max}$, then the algorithm is terminated. We observe in both tables that the HEA versions find global solutions in a relatively small number of function evaluations (N_f), and after that, the algorithms were still running until one of the termination conditions is met in order to check whether there are any other solution left or not.

Table 2 reveals that the HEA_1 finds no less than N_{gmax} (=20) global solutions for five test problems. In fact, we may conclude that these problems have infinitely many solutions, and by setting N_{gmax} bigger, it is possible to find as many solutions as one may want. For the other five problems, the algorithm was terminated because the number of ineffective local transformations exceeded $N_{max}(= 10)$.

Table 2: Numerical Results for the HEA with Population Update Rule 1.

Problem	n	K_{min}	K_{av}	K_{max}	N_{gen}	N_{loc}	NF	N_f
badfree	5	20	20	20	31	14	3260	3260
games	16	20	20	20	74	432	32859	32859
kojshin	4	2	2	2	38	115	4023	1474
mathinum	3	1	14.4	20	255	858	30402	23066
mathisum	4	1	1.9	2	52	184	6751	2974
ne-hard	3	2	3.1	4	185	956	31223	16068
powell	16	4	11.4	20	96	1644	45740	36676
powell_mcp	8	2	5.1	9	110	1109	24585	10553
scarfasum	14	2	2.7	3	80	1157	45508	25139
sppe	27	20	20	20	325	471	138475	138475

Table 3 shows that, the performance of the HEA_2 is promising except for problem *mathinum*, for which it occasionally failed to find a global solution despite the fact that this problem has at least twenty solutions as shown in Tables 2 and 3. It happened because the number of ineffective local transformations reached its limit $N_{max}(= 10)$ before finding a global solution. By increasing N_{max}, we could improve the performance of the HEA_2 for this problem. Another possible reason for the unexpected performance of the HEA_2 for problem *mathinum* is the choice of parameter ε_D. As we mentioned earlier in Section 3.1, the Population Update Rule 2 is an extension of the standard genetic algorithm selection mechanism and it tries to prevent the population from prematurely converging to one or only a few points. However, we found that the choice $\varepsilon = n/5$ was not appropriate for problem *mathinum*, since the HEA_2 could not escape from the undesirable property that the population converges to only a few points. We have observed that, by increasing ε_D, we could improve the performance of the HEA_2 for this problem.

Table 3: Numerical Results for the HEA with Population Update Rule 2.

Problem	n	K_{min}	K_{av}	K_{max}	N_{gen}	N_{loc}	NF	N_f
badfree	5	20	20	20	15	0	2946	2946
games	16	20	20	20	31	132	25529	25529
kojshin	4	2	2	2	28	92	5786	1986
mathinum	3	0	4.3	20	794	1008	132280	41290
mathisum	4	1	1.95	2	48	95	9299	2728
ne-hard	3	2	3.3	4	82	261	17930	10746
powell	16	5	14	20	71	1160	49965	34619
powell_mcp	8	3	6.9	11	107	1123	32349	12887
scarfasum	14	1	1.6	3	49	714	55479	22456
sppe	27	20	20	20	225	1563	260650	260650

Finally, we make some remarks on the comparison between the results shown in Tables 2 and 3 in terms of the numbers of obtained global solutions and computational costs. Generally, the HEA versions are neutral in terms of the numbers of obtained global solutions, since these numbers are almost the same for six problems out of ten. For problems *powell* and *powell_mcp*, the HEA_2 was able to find more global solutions than the HEA_1. However, for problems *mathinum* and *scarfasum*, the HEA_1 performed better than the HEA_2 in terms of the numbers of obtained global solutions and the numbers of function evaluations. On the other hand, the

HEA_2 did not use local search in all runs for problem *badfree*, while the HEA_1 required more local search steps than the HEA_2 for seven out of ten problems.

5 Conclusions

In this paper, we have presented a new population-based algorithm HEA that is designed to find as many solutions as possible of the general VIP. New types of population update schemes in the evolutionary algorithm and a hump-tunneling technique for escaping from detected solutions have also been proposed. The computational results for some well known test problems show that the HEA method is capable of locating many solutions in an acceptable number of function evaluations. Moreover, the numerical results indicate that, the more solutions a problem has, the better the HEA method works. Finally, it is worth mentioning that one can use the HEA for finding solutions of a system of equations or a global optimization problem with known minimum objective value.

Acknowledgement. The authors are grateful to Professor R. Enkhbat of National University of Mongolia for suggesting a global optimization approach to the general variational inequalities.

References

[1] Al-Khayyal, F.A. (1987), An implicit enumeration procedure for the general linear complementarity problem, *Mathematical Programming Study*, 31, 1–20.

[2] De Jong, K.A. (2005), *Evolutionary Computation*, The MIT Press, Cambridge, MA.

[3] Facchinei, F. and Pang, J.S. (2003), *Finite-Dimensional Variational Inequalities and Complementarity Problems*, Volumes I and II, Springer-Verlag, New York.

[4] Fukushima, M. (1996), Merit functions for variational inequality and complementarity problems, In: Di Pillo, G. and Giannessi, F. (eds.), *Nonlinear Optimization and Applications*, Plenum Press, New York, pp. 155–170.

[5] Ge, R. (1990), A filled function method for finding a global minimizer of a function of several variables, *Mathematical Programming*, 46, 191–204.

[6] Goldberg, D.E. (1989), *Genetic Algorithm in Search, Optimization and Machine Learning*, Addison-Wesley, Reading, Mass.

[7] Herrera, F., Lozono, M. and Verdegay, J.L. (1998), Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis, *Artificial Intelligence Review*, 12, 265–319.

[8] Kanzow, C. (2000), Global optimization techniques for mixed complementarity problems, *Journal of Global Optimization*, 16, 1–21.

[9] Kanzow, C. and Pieper, H. (1999), Jacobian smoothing methods for nonlinear complementarity problems, *SIAM Journal on Optimization*, 9, 342–373.

[10] Kostreva, N.M. and Zheng, Q. (1994), Integral global optimization method for solution of nonlinear complementarity problems, *Journal of Global Optimization*, 5, 181–193.

[11] Laguna, M. and Marti, R. (2003), *Scatter Search: Methodology and Implementation in C*, Kluwer Academic Publishers, Boston.

[12] Laguna, M. and Marti, R. (2005), Experimental testing of advanced scatter search designs for global optimization of multimodal functions, *Journal of Global Optimization*, 33, 235–255.

[13] Levy, A.V. and Montalvo, A. (1985), The tunneling algorithm for the global minimization of functions, *SIAM Journal on Scientific and Statistical Computing*, 6, 15–29.

[14] Pardalos, P.M. and Rosen, J.B. (1988), Global optimization approach to the linear complementarity problem, *SIAM Journal on Scientific and Statistical Computing*, 6, 341–353.

[15] Qi, H.D. and Liao, L.Z. (2000), A smoothing Newton method for general nonlinear complementarity problems, *Computational Optimization and Applications*, 17, 231–253.

[16] Rago, C. and Alidaee, B. (2005), *Metaheuristics Optimization via Memory and Evolution*, Kluwer Academic Publishers, Boston, MA.

[17] Selami, H. and Robinson, S.M. (1997), Implementation of a continuation method for normal maps, *Mathematical Programming*, 76, 563–578.

[18] Talbi, E. (2002), A taxamony of hybrid metaheuristics, *Journal of Heuristics*, 8, 541–564.

[19] Xu, Z., Huang, H.X., Pardalos, P.M. and Xu, C.X. (2001), Filled functions for unconstrained global optimization, *Journal of Global Optimization*, 20, 49–65.