Simulation-based calibration of geotechnical parameters using parallel hybrid moving boundary particle swarm optimisation

Y. Zhang[1], D. Gallipoli[2] and C.E. Augarde[1]

[1]School of Engineering, Durham University, Durham, UK

[2]Department of Civil Engineering, University of Glasgow, Glasgow, UK

Abstract

Simulation-based optimization methods have been recently proposed for calibrating geotechnical models from laboratory and field tests. In these methods, geotechnical parameters are identified by matching model predictions to experimental data, i.e. by minimizing an objective function that measures the difference between the two. Expensive computational models, such as finite difference or finite element models, are often required to simulate laboratory or field geotechnical tests. In such cases, simulation-based optimization might prove demanding since every evaluation of the objective function requires a new model simulation until the optimum set of parameter values is achieved. This paper introduces a novel simulation-based "hybrid moving boundary particle swarm optimization" (hmPSO) algorithm that enables calibration of geotechnical models from laboratory or field data. The hmPSO has proven effective in searching for model parameter values and, unlike other optimization methods, does not require information about the gradient of the objective function. Serial and parallel implementations of hmPSO have been validated in this work against a number of benchmarks, including numerical tests, and a challenging geotechnical problem consisting of the calibration of a water infiltration model for unsaturated soils. The latter application demonstrates the potential of hmPSO for interpreting laboratory and field tests as well as a tool for general back-analysis of geotechnical case studies.

Keywords: parameter identification, unsaturated soils, particle swarm optimization, parallel computing

INTRODUCTION

During the last decade, research has been carried out on the use of optimization methods for the identification of geotechnical parameters from both laboratory tests [1] and field tests (e.g. pressuremeter tests) [2]. Optimization methods require the use of relevant engineering models [3] to simulate a particular test for which measurements are available. For laboratory tests, it is possible to use relatively simple numerical or analytical models to reproduce soil behaviour at stress-point level when a uniform distribution of stresses and strains can be reasonably assumed. On the other hand, more complex models (such as finite element or finite difference models) are necessary for field tests because of the non-uniform stress and strain fields generated within the soil domain. In the latter case, the use of optimization methods might therefore prove computationally demanding because a new model simulation must be performed for every single evaluation of the objective function until the optimum set of parameter values is achieved.

In this paper, global optimization methods are described for minimizing nonlinear objective functions over a bounded search space. Mathematically, the problem is expressed as

$$\text{minimize} \quad f(\mathbf{x}) \tag{1}$$

$$\text{subject to} \quad \mathbf{x} \in D^n \text{ and } \mathbf{x}_l \leq \mathbf{x} \leq \mathbf{x}_u$$

In Eqn 1 $\mathbf{x}$ is the vector $[x_1, x_2, \ldots, x_n]^T$ located within the range $[\mathbf{x}_l, \mathbf{x}_u]$ of the n-dimensional space D^n, $\mathbf{x}_l$ and $\mathbf{x}_u$ are the lower and upper search bounds respectively and $f(\mathbf{x})$ is the objective function, which is calculated either from a closed-form mathematical solution or (as in this case) from a computer simulation of a real engineering problem for which measurements are available.

For geotechnical optimization problems, the aim is to obtain a set of model parameter values that provide the best match between model simulations and measurements. Interest does not lie in finding the absolute global minimum of the objective function, as this merely confirms a close match between simulation and reference measurements, but rather in finding the global minimum within a restricted search space corresponding

to the range of realistic parameter values for engineering design.

Moreover, the complexity of current engineering models can often lead to the existence of several local minima within the search space, each with very similar values of the objective function despite large differences in some of the associated parameter values. The number of local minima and the corresponding values of the objective function also depend on the particular computational model adopted for the simulations as well as on the accuracy of the calculations (which can sometimes lead to “noise” in the objective function). Simulation-based calibration of engineering parameters therefore requires the combination of a global search strategy capable of coping with several local minima together with accurate robust computational models.

Evolutionary Algorithms (EAs) are population-based iterative stochastic algorithms that have proved effective in many areas of engineering for solving complex optimization problems where the objective function has a large number of local minima. EAs are based on Darwinian theories simulating the evolutionary process of a population of individuals towards the best position (i.e. the final solution of the optimization problem) by competition and natural selection. One of the best known EAs is the Genetic Algorithm (GA), where the optimization process consists in finding the best position iteratively through operations, such as crossover and mutation, on a population of individuals and by application of rules for the selection of individuals into subsequent generations. Particle Swarm Optimization (PSO) is a newer evolutionary technique inspired by the observed social behaviour of bird flocks and fish schools and was first introduced by Kennedy and Eberhart [4]. More recently, PSO has been shown to have increased effectiveness and better computational efficiency over other EAs while maintaining simplicity [5-8]. In contrast to the GA, PSO has no evolution operators such as crossover and mutation which makes it ideal for asynchronous parallel implementation [16], an important issue for simulation-based optimization where parallel implementation is necessary as discussed later. An additional advantage of PSO over other EAs is the ease with which it can be combined with other search algorithms.

Although PSO is effective in finding a global optimum, it suffers from what is termed “premature convergence” when the optimization process prematurely converges to a local optimum because it is no longer able to explore other areas of the search space where better positions or the global best position may lie [9]. Another weakness is that

PSO searching ability reduces significantly in the later stages of optimisation [10] and this sometimes leads to slow convergence rates.

To overcome such known weaknesses of PSO, a hybrid optimisation approach can be adopted. A local search method can be incorporated into PSO to carry out quick and efficient explorations around potential optima at a much lower computational cost [30]. Local search methods tend to find the best solution around the neighbourhood of the starting point with a better rate of convergence than PSO in the same situation. However, local search methods are rarely the best approach to explore the whole space of potential solutions, hence the need of combining them with an effective global technique like PSO.

Hybridization of a global search method with a local search method has proved very effective in solving optimization problems by making good use of the strengths of both global and local methods [5, 13-15]. For example, performance of GAs has often been improved through hybridization with a local search method [29] following two methodologies. The first is to use the local search technique during the evolutionary loop, i.e. local searches are regularly carried out in parallel with the evolutionary algorithm to accelerate convergence rate. The second is to use the local search technique at the end of the evolutionary loop, i.e. one local search is carried out using results from the GA as initial guesses to improve the solution. Similar ideas have also been applied to PSO [30, 31]

In this paper we describe the development and use of a hybrid global-local optimization method particularly suited to the calibration of geotechnical models by using finite element simulations of real engineering problems. The proposed algorithm differs from other proposals in two main aspects: firstly a moving-boundary PSO method is employed for the global search and secondly an approximate local search algorithm is used instead of a complete local search. We present details of the proposed hybrid optimization method together with its implementation in both serial and parallel environments. The performance of the algorithm is also demonstrated with specific reference to the determination of geotechnical parameters

PARTICLE SWARM OPTIMIZATION

The original PSO algorithm was introduced as a population-based stochastic global optimization method by Kennedy and Eberhart [4, 17]. It is inspired by the observed

social behaviour of “swarms” in which a group of independent “individuals” or “particles” try to achieve a goal by acting in a complex and coordinated way. As for the GA, PSO starts by randomly initializing a population of individuals in the search space. The exploration for the best solution then continues with particles changing through successive “generations” (a generation is an iteration of the algorithm corresponding to one update of the whole population) according to rules until a termination criterion is met. The basic PSO used in this paper is denoted as “bPSO” and was introduced by Shi and Eberhart [18] as a variant of the original PSO.

Before presenting the detailed algorithm, some notation is defined. The size of the swarm population is N_p. The current position of the ith particle is represented by the n-dimensional vector $\mathbf{x}_i$ in the search space and by the corresponding fitness $f(\mathbf{x}_i)$ (this is the “quality” of the particle’s position measured by the value of the objective function). Each particle also has a “velocity” $\mathbf{v}_i$, which measures the change of the particle’s position in each generation. The best position achieved so far by the ith particle *pbest* is identified by the point $\mathbf{P}_i$ and the corresponding fitness $f(\mathbf{P}_i)$. The best position achieved so far by the whole swarm *gbest* is identified by the point $\mathbf{P}_g$ and the corresponding fitness $f(\mathbf{P}_g)$. Each particle has therefore a “memory” of its own best position as well as a “social” knowledge of the best position achieved by neighbouring particles. With specific reference to geotechnical applications, each vector $\mathbf{x}_i$ contains a different set of parameter values for an individual model simulation and the search space is the space of all model parameters.

The PSO starts by randomly initializing each particle’s velocity $\mathbf{v}_i$ and position $\mathbf{x}_i$ and by calculating the corresponding fitness $f(\mathbf{x}_i)$. After initialization, particles “fly” around in the search space for many generations seeking for the optimum fitness, i.e. the minimum of the objective function (in each generation the algorithm updates both velocity and position of all particles within the swarm). The rule for updating velocity is:

$$\mathbf{v}_i^{k+1} = w^k \mathbf{v}_i^k + c_1 r_1 (\mathbf{P}_i - \mathbf{x}_i) + c_2 r_2 (\mathbf{P}_g - \mathbf{x}_i) \qquad (2)$$

where the superscripts k, $k+1$ etc. indicate the generation number, r_1 and r_2 are two random factors in the interval [0,1], w^k is the inertia weight and c_1 and c_2 are the "cognitive" and "social" weights respectively (default values for most problems are c_1 = c_2 = 2.0 [20]). To avoid the algorithm becoming divergent, the particles' velocities are

confined to a range $-\mathbf{v}_{max} < \mathbf{v}_i < +\mathbf{v}_{max}$ where the limit velocity $\mathbf{v}_{max}$ determines the maximum change of position for a given particle. The parameter $\mathbf{v}_{max}$ is usually set equal to half the width of the search domain to cover the span of the parameter space. Updated particles' positions at generation $k+1$ are obtained by incrementing the position at generation k with the velocity at $k+1$.

The inertia weight w^k [19] controls the "momentum" of the particle. A large value of the inertia weight favours global exploration by searching new areas, while a small value favours local exploration. A linear decrease of inertia weight with iterations is introduced to focus in on a global minimum [19]:

$$w^k = (w_{max} - w_{min}) * (MaxIter - k) / MaxIter + w_{min} \quad (3)$$

where w^k is the inertia weight for the k^{th} iteration, $MaxIter$ is the maximum iteration number and w_{max} and w_{min} are the maximum and minimum inertia weights set by the user.

LOCAL SEARCH METHOD

Among the variety of local search methods available in the literature, the Nelder-Mead (NM) method [12] has been chosen in this research. Similarly to the bPSO used for the global search, the NM method is a "direct" algorithm, i.e. it requires evaluation of the objective function at different points within the search space but does not require information about the gradient of the objective function. This feature is particularly useful for simulation-based optimisation problems, such as the geotechnical applications considered here. For these problems, gradient measures are difficult to obtain from computational models, e.g. a finite difference or finite element simulation of an engineering boundary value problem will give no information about how results change when model parameters are varied.

The NM method requires the definition of $n+1$ starting points in the search space, where n is the dimension of the search space. These points form the initial vertices of a working simplex, which is then subjected to a series of "transformations" aimed at decreasing the values of the objective function at its vertices. Such transformations are governed by four operations controlling reflection, contraction, expansion and shrinkage of the simplex. The simplex must also remain non-degenerate throughout

the series of transformations. This means that, if any of the $n+1$ vertices is taken as the origin, the n vectors connecting the origin with the remaining vertices must span the n-dimensional space or, in other words, the vertices of the simplex must not lie in the same hyperplane. The sequence of transformations is terminated when the size of the working simplex or the difference between the values of the objective function at its vertices become smaller than a given tolerance. Full details of the NM method can be found in [12].

A HYBRID MOVING-BOUNDARY PSO WITH APPROXIMATE LOCAL SEARCH

Since the introduction of the original PSO, various improvements have been proposed to reduce difficulties associated to parameter selection [21, 22], to avoid premature convergence [23-27] and to increase computational speed [28]. The increase of computational speed is achieved in this work while keeping swarm diversity to avoid premature convergence, by proposing a new hybrid moving-bounds PSO algorithm (hmPSO) that combines the effectiveness of the bPSO and NM methods for global and local searches respectively.

The new algorithm operates in a normalized search space. Normalization is particularly important for geotechnical applications where model parameter values can vary over ranges of very different magnitude. For example, permeability can range from 10^{-3} to 10^{-9} ms^{-1} while elastic bulk modulus can range from 10^{4} to 10^{7} Pa. Normalization therefore facilitates comparison between all parameter scales and enables calculation of normalized “distances” in the search space. Such calculation is frequently performed during the optimization process and, if the dimensions of the search space are of disparate sizes, the algorithm will result in errors.

Moreover, for those parameters whose values evenly span several orders of magnitude, logarithmic normalisation of the search range will allow an efficient random initialization of particle positions by obtaining a uniform spread across the different orders of magnitude. Given that PSO has a stochastic basis, the quality of the initial random distribution of particle positions over the search space is crucial to the performance of the algorithm.

Linear normalization of a given parameter range involves linear mapping between the original search range and a scaled domain. In particular, the *j*th component of the of the *i*th particle position $\mathrm{x}_{\mathrm{i}\,j}$ is normalized by the following relationship:

$$\overline{\mathrm{x}}_{\mathrm{i}\,j} = \frac{\mathrm{x}_{\mathrm{i}\,j} - \mathrm{x}_{\mathrm{l}\,j}}{\mathrm{x}_{\mathrm{u}\,j} - \mathrm{x}_{\mathrm{l}\,j}} \left(Nmax - Nmin\right) + Nmin \tag{4}$$

where $\overline{\mathrm{x}}_{\mathrm{i}\,j}$ is the *j*th normalized component the *i*th particle position, $\mathrm{x}_{\mathrm{u}\,j}$ and $\mathrm{x}_{\mathrm{l}\,j}$ are the *j*th components of the upper and lower bounds of the search space while $Nmax$ and $Nmin$ are the normalized upper and lower bounds (assumed to be the same for all components).

Logarithmic normalization of a given parameter range involves linear mapping between the logarithmic representation of the original search range and a scaled domain. By using similar definitions as in Eqn 4, the *j*th component of the *i*th particle position $\mathrm{x}_{\mathrm{i}\,j}$ is logarithmically normalized by using the following relationship:

$$\overline{\mathrm{x}}_{\mathrm{i}\,j} = \frac{\mathrm{x}'_{\mathrm{i}\,j} - \mathrm{x}'_{\mathrm{l}\,j}}{\mathrm{x}'_{\mathrm{u}\,j} - \mathrm{x}'_{\mathrm{l}\,j}} \left(Nmax - Nmin\right) + Nmin \tag{5}$$

where this time the logarithms of the *j*th components are used in the normalization, i.e.

$$\mathrm{x}'_{\mathrm{i}\,j} = \log \mathrm{x}_{\mathrm{i}\,j} \quad , \qquad \mathrm{x}'_{\mathrm{l}\,j} = \log \mathrm{x}_{\mathrm{l}\,j} \quad , \qquad \mathrm{x}'_{\mathrm{u}\,j} = \log \mathrm{x}_{\mathrm{u}\,j} \tag{6}$$

In the following it is assumed that all dimensions of the search space are normalized (either using linear or logarithmic scaling as appropriate for each parameter) between the same lower and upper bounds N_{min}=1 and N_{max}=2 respectively.

The sequential hmPSO algorithm

The hybrid moving-bounds hmPSO algorithm consists of three main components: a global bPSO, a local bPSO and a direct local search performed by the NM method.

The global bPSO operates over the whole search domain and employs a “global swarm” that is initialized only once at the start of the optimization process.

The local bPSO operates over a smaller search sub-domain centred on the current best particle position and employs a “local swarm”, whose particle positions are initialized every time the boundaries of the search sub-domain are updated. By using a contracted search range, the efficiency of the bPSO algorithm improves significantly [32].

The use of global and local swarms allows simultaneous exploration of the search space and exploitation of the most promising search regions. The global swarm investigates the original search space to maintain the diversity of the population while the local swarm focuses on a smaller search space to increase efficiency. Particles fly within their respective search spaces as during normal bPSO except that particles in the local swarm fly over a smaller search sub-domain. Exchange of “social” knowledge between the two swarms is also ensured by sharing the same best particle position *gbest*.

The Simplex routine is called every N_L bPSO generations (note that each iteration includes updates of all particles in the global and local swarms) to undertake a fast exploration of the sub-region containing the best $n+1$ particle positions, where n is the dimension of the search space. If the best $n+1$ particle positions form a non-degenerated simplex, they are used as initial vertices. Otherwise, a regular simplex is created starting from the centroid of a sub-set of best particle positions (in this work the centroid of the best $N_p/4$ particle positions is used).

The solution $\mathbf{x}_L$ corresponding to the smallest value of the objective function found by the NM method is taken as the centre of the updated search sub-domain $[\mathbf{x}_L-\delta\mathbf{I}\ ,\ \mathbf{x}_L+\delta\mathbf{I}]$ explored by the local bPSO. The size of the search sub-domain is given by the scalar radius δ multiplied by the n-dimensional unit vector $\mathbf{I}$.

The convergence rate of the NM method is very sensitive to the quality of the starting points and is significantly improved by using the best $n+1$ particle positions provided by the bPSO as the initial vertices of the simplex. Similarly, premature convergence and low efficiency of bPSO are overcome by alternating sequences of bPSO generations with local exploration of the most promising areas of the search space.

A schematic representation of the algorithm is given in Figure 1 where k is the iteration number, *particleIDs* are the labels of particles allocated to the local bPSO search (these are chosen beforehand) and I_{L} is the counter of local searches.

In comparison with the original bPSO, the hmPSO requires the following sequence of additional computations:

- sorting particles in accordance with their best position $\mathbf{P}_{\mathrm{i}}$;
- performing a NM local search;
- initializing the local swarm over the updated search sub-domain $[\mathbf{x}_{\mathrm{L}}\text{-}\delta\mathbf{I}\ ,\ \mathbf{x}_{\mathrm{L}}\text{+}\delta\mathbf{I}]$.

These three additional computations are carried out every N_{L} generations of the bPSO.

The focus of the local bPSO is progressively restricted during the optimization process by decreasing the radius δ of the search sub-domain as the number of local searches I_{L} increases. The radius δ starts from a relatively high value of 0.4 and then decreases linearly with the number of local searches until reaching a limit value of 0.2, corresponding to the maximum number of local searches set by the user. It is useful recalling here that the radius of the entire normalized search space is equal to 1.0.

In hmPSO, the NM local search is halted after a maximum number of transformations even if the termination criterion is not met. This tends to happen especially for the first few local searches as the initial guesses of the simplex vertices are poor and probably misleading. Such earlier termination of the algorithm can save significant computational time and avoids getting “trapped” into minima.

The parallel hmPSO algorithm

EAs are renowned for being computationally expensive and the bPSO algorithm typically requires thousands of generations to converge. Each generation involves N_p evaluations of the objective function and hence, for the simulation-based optimization process considered here, N_{p} model simulations.

Parallel implementation is an appealing means of increasing algorithmic efficiency by assigning different particles to different processors working simultaneously. Parallel implementation of bPSO can either be “synchronous” or “asynchronous”. In the

“synchronous” implementation, generations are carried out sequentially and a new generation is started only after the previous generation is completed. This means that each particle must wait for all other particles to update their positions before moving to the next generation. In the “asynchronous” implementation the idea of sequential generations is abandoned as each particle position is continuously updated (regardless of the status of neighbouring particles) while an uninterrupted sequence of local searches is also simultaneously executed.

In a simulation-based optimization process, each simulation may take a different amount of time depending on parameter values and model non-linearity. This will result in load unbalances among processors and, in a synchronous parallel implementation, the slowest processor will determine the speed of the algorithm.

Unlike GAs, for which the synchronous parallel implementation is the only option (due to synchronized communication between individuals during crossover and mutation), bPSO lends itself to asynchronous parallel implementation. In bPSO, particles only share the “social” knowledge of the best position $gbest$ and this single piece of information is easily distributed by using a client-server model, where each particle queries a central store of shared data. For distributed computing, clients and servers are independently placed on network nodes, which may also use different hardware and operating systems.

The parallel server-client implementation of the hybrid moving-bounds hmPSO algorithm is schematically explained by Figure 2. The server-client model consists of one server and a number of clients divided in two distinct groups, i.e. the particle client group and the local search client group. Nodes in the particle client group request information directly from the server and they do not communicate among themselves. On the other hand, nodes in the local search client group interact according to a master-slave model. The master is responsible for managing and coordinating all slaves in the group while the server only communicates with the master of the local search client group. In this implementation, a single processor has been dedicated to NM local searches so the total number of processors N_p+2, i.e. one server processor, N_p processors for N_p particles and one processor for the local searches.

The server is responsible for storing and managing shared data, listening to queries by all clients and returning the relevant responses. The clients are responsible for

evaluating the objective function at each particle position (particle client group) or performing a local search (local search client group).

The flowchart of the whole algorithm is illustrated in Figure 3 where dotted lines indicate the flow of data. The parallel programming library MPI [35] is used for the data communications among processors. The algorithm is started by initializing particle clients whose position is then communicated to the server. Subsequently, the algorithm advances as the server continues receiving and parsing different types messages from both particle and local search clients.

When the server receives the best position *pbest* of a given client particle, a check is performed whether the swarm best position *gbest* has changed and an updated *gbest* is sent back to particle clients for calculating the next position. When the server receives a local search solution $\mathbf{x}_L$, a check is again performed whether the swarm best position *gbest* has changed. Subsequently, as particles best positions *pbest* are continuously updated by bPSO, the server asks the client to perform the next local search based on fresh improved initial guesses.

Local searches are used to guide the local swarm towards global optimum and, when a new local search solution $\mathbf{x}_L$ becomes available, the focus of the bPSO search sub-domain is shifted to the region around such solution. At the same time, a request is sent by the server to the client particles of the local swarm to reinitialize their positions and velocities over the new search sub-domain. If the server receives a request to stop from a client (either a particle or a local search client) due to the attainment of a minimum of the objective function within the set tolerance, it sends an order to terminate the algorithm to all clients.

Note that, in order to ensure that reasonable initial guesses are used for the very first local search, a preset number of bPSO evaluations must be carried out by particle clients before local search clients become active.

ASSESSMENT OF ALGORITHM PERFORMANCE

In this section, the serial and parallel implementations of the hmPSO algorithm are assessed for a number of benchmark tests. The benchmarks include five numerical tests based on artificial objective functions, with well defined mathematical forms, as well as one simulation-based geotechnical optimization test consisting in the calibration

of an unsaturated soil infiltration model. The five numerical tests are used to assess robustness and efficiency of the (serial) hmPSO algorithm with respect to the standard bPSO algorithm while the simulation-based optimization compares the performances of the parallel and serial implementations hmPSO.

The termination criterion should be set to end computations when the global minimum of the objective function is found within a given tolerance. In real optimization problems, however, this is not easy task since the global minimum of the objective function is not known in advance. For example, in the engineering application to the identification of model parameters, the objective function is given by the difference between model simulations and measurements from laboratory or field tests; the smaller such difference, the more accurate the evaluation of model parameters.

In the benchmark tests presented in this work the global minimum of the objective function is known a priori (and is equal to zero as shown later) so the definition of a suitable termination tolerance is easier than for the general optimization problem. Other termination criteria used in PSO include ending computations when the smallest value of the objective function remains unchanged over a certain number of generations and when the number of function evaluations exceeds a maximum value set by the user. This maximum value usually depends on type and dimension of the particular optimization problem and should be set to avoid termination when convergence might still be achieved.

Numerical tests

Numerical tests are used to assess the robustness and efficiency of hmPSO in finding the global minimum of five mathematical functions. The characteristics of such functions, which have been previously employed in the literature to test optimisation algorithms [34], are summarized in Table 1. Inspection of Table 1 indicates that four of the five test functions are multimodal, for which the search of the global minimum can prove particularly challenging because the large number of local minima increases the likelihood of premature convergence. To ensure that the algorithm is tested under the most general conditions, four out of five functions are shifted so that their respective global optima do not lie at the centre of the search space. In addition, for each function, the search is performed in both five-dimensional and ten-dimensional spaces, where the range of variation for each dimension is kept the same (see Table 1).

These numerical tests follow the procedures set by the 2005 IEEE Congress on Evolutionary Computation [34], which defined methods for evaluating algorithmic efficiency and robustness. To measure the performance of the algorithms accurately, the search for the global minimum of each function is repeated 25 times by using exactly the same set of parameters as given in Table 2. Indices measuring algorithm performance are therefore defined based on such 25 runs in a statistical sense so as to take randomness into account.

For each run, the objective function $f(\mathbf{x})$ is defined as:

$$f(\mathbf{x}) = abs\left[g(\mathbf{x}) - g(\mathbf{x}^*)\right] \tag{7}$$

where $g(\mathbf{x})$ is the chosen test function, $g(\mathbf{x}^*)$ is the global minimum of such test function and *abs* indicates the absolute value. A run is considered successful if the value of the objective function drops below a termination tolerance of 1.0e-5 within a maximum number of function evaluations equal to 10000 times the test dimension. Robustness is measured by the "success rate" over 25 runs, which is compared in Table 3 for both algorithms and all numerical tests. It can be seen that hmPSO significantly outperforms bPSO by a notably higher success rate in all tests with the only exception of the Rasgrigin function. In several tests, bPSO also failed to find the global minimum while hmPSO succeeded to find it. The better performance of hmPSO with respect to bPSO is also confirmed by Figure 4, which provides histograms of the mean values (in a logarithmic scale) of the objective function calculated by the two algorithms over 25 runs for each numerical test. In Figure 4 the largest positive and negative deviations from such mean values are also given as error bars.

Similarly, Figure 5 shows histograms of the mean number of function evaluations (in a logarithmic scale) performed by both algorithms over a total of 25 runs for each numerical test. Again, the largest positive and negative deviations from such means are given in Figure 5 as error bars. It is worth noting that, for Shifted Rosenbrock (n=5), Shifted Rosenbrock (n=10) and Shifted Griewank (n=10), the number of bPSO function evaluations is constant for all 25 runs. This is because, for these three numerical tests, the bPSO algorithm failed to converge in all runs and the number of function evaluations was always equal to the maximum limit (i.e. 10000 times the test dimension). Inspection of Figure 5 indicates that hmPSO shows significantly greater

efficiency than bPSO by using a smaller number of function evaluations in all numerical tests.

Figure 6 illustrates the convergence rate of both algorithms for the Shifted Rosenbrock (n=10) numerical test and provides graphical evidence of the dramatically better performance of hmPSO in comparison with bPSO. The sequence of NM local searches improves the solution in a step-wise fashion as evident from the jumps in the hmPSO curve of Figure 6, which correspond to the availability of new solutions from local searches. It is also worth pointing out that, if the NM method is used on its own without combination with bPSO, all numerical tests fail to converge.

Simulation-based optimization test

The above numerical tests demonstrate that the serial version of the hmPSO algorithm is both robust and efficient. This section investigates the application of the hmPSO algorithm to the identification of parameter values for the specific geotechnical model of one-dimensional water infiltration in an unsaturated soil column with a rigid soil skeleton. One of the challenges of such application is the lack of a priori knowledge about the nature of the search space and, in particular, whether the objective function presents a large or small number of local minima.

This application is also used to compare the performances of the serial and parallel implementations of hmPSO where, in the latter, each processor is allocated to a different particle as previously described. The hardware for the parallel implementation of hmPSO is a Linux cluster "Hamilton" at Durham University. The cluster consists of 96 dual-processor dual-core Opteron 2.2 GHz nodes with 8 GBytes of memory and a Myrinet fast interconnect for running MPI code as well as 135 dual-processor Opteron 2 GHz nodes with 4 GBytes of memory and a Gigabit interconnect. The operating system is SuSE Linux 10.0 (64-bit) and disk storage capacity is 3.5 Terabytes.

Water flow is modelled according to the "θ-based" form of Richards' equation [36]:

$$\frac{\partial \theta}{\partial t} = \frac{\partial}{\partial z}\left[D(\theta) \frac{\partial \theta}{\partial z} \right] + \frac{\partial K(\theta)}{\partial z} \qquad (8)$$

where $K(\theta)$ is the unsaturated hydraulic conductivity (m/s), θ is the volumetric water content, $D(\theta)$ is the unsaturated hydraulic diffusivity (m^2/s), t is time (s) and z is the

vertical distance (m) measured positive upwards. The initial and boundary conditions are expressed as:

$$h(z,0) = h_o \qquad 0 < z < L \tag{9}$$

$$h(0,t) = h_b \qquad t > 0 \tag{10}$$

$$h(L,t) = h_t \text{ or } q(L,t) = q_t \qquad t > 0 \tag{11}$$

where L is the height of the soil column (m), h and q are the pressure head and flux (m and m/s) respectively, h_o is the initial pressure head distribution across the column, h_b and h_t are the constant pressure heads at the bottom and top of the soil column respectively and q_t is the constant flux imposed at the top of the soil column.

The Mualem-van Genuchten model [36, 37] is employed to describe the soil water retention properties in which the relationship between the effective degree of saturation S_e and pressure head h is given by:

$$S_e = \left[1 + |\alpha h|^{\beta}\right]^{-m} \tag{12}$$

where α (>0, m^{-1}) is a parameter related to the inverse of the air-entry pressure, β (>1) is a dimensionless parameter related to the pore size and m is a dimensionless parameter depending on β through the relationship $m=1-1/\beta$.

The effective saturation S_e is given by:

$$S_e = (\theta - \theta_r)/(\theta_s - \theta_r). \tag{13}$$

where θ_s is the saturated volumetric water content and θ_r is the residual volumetric water content;

The unsaturated hydraulic conductivity K is given as the product of the saturated hydraulic conductivity K_s by the dimensionless relative permeability k_r . The relative permeability, which is smaller than unity, accounts for partial saturation through the following dependency on the effective degree of saturation:

$$K = K_s k_r = K_s S_e^{1/2}\left[1-\left(1-S_e^{1/m}\right)^m\right]^2 \tag{14}$$

The unsaturated diffusivity $D(\theta)$ can be derived as:

$$D = K\frac{dh}{d\theta} = \frac{K_s(1-m)}{\alpha m(\theta_s-\theta_r)} S_e^{1/2-1/m}\left[A^{-1}+A-2\right] \tag{15}$$

where $A=(1-S_e^{1/m})^m$.

The determination of the five parameters β, α, θ_s, θ_r and K_s through simulation-based optimization of an infiltration model has been shown to be a challenging problem [38]. This problem is here used as a benchmark test for the hmPSO algorithm where experimental measurements are replaced by the results computed from a finite difference model with known parameter values (the "forward" analysis). In the forward analysis, the height of the unsaturated soil column is assumed as L = 1.0 m and the parameter values are chosen as α = 3.35m^{-1}, β = 2, K_s = 9.22E-5 m/s, θ_s = 0.368 and θ_r = 0.102. These are realistic parameter values corresponding to a field site in New Mexico (see Celia et al. [39]). The chosen initial and boundary conditions are:

$$h(z,0) = -10\,\text{m} \qquad 0 < z < L \tag{16}$$

$$h(0,t) = -10\,\text{m} \qquad t > 0 \tag{17}$$

$$h(1.0\,\text{m},t) = -0.75\,\text{m} \qquad t > 0 \tag{18}$$

In the finite difference simulation, the duration of the infiltration process is 6 hours and this has been divided in equal time steps of 36 seconds while the soil column was discretised using 100 elements along the vertical direction. The computed profiles of pressure head and water content at different times are illustrated in Figures 7 and 8.

The test consists in searching through a five-dimensional space for the vector $\mathbf{x}=[\alpha, \beta, K_s, \theta_s, \theta_r]^T$ whose components are the model parameter values providing an optimum match to the data in Figures 7 and 8 (as if these data were experimental measurements). Given that these data can be perfectly matched by the model, the aim is to assess the ability and efficiency of the hmPSO algorithm in returning the same set

parameter values of the forward analysis. Table 4 shows the search ranges used in this optimization test for each of the five model parameters.

Table 5 provides the algorithmic settings for the hmPSO, where the maximum number of evaluations of the objective function was set at a relatively high value of 500000 considering the difficulties of this particular benchmark and to allow a reasonable comparison of the serial and parallel implementations of hmPSO.

The following objective function (similar to that used in [40]) is minimized in this optimization test:

$$f(\mathbf{x}) = \sqrt{\sum_{j=1}^{p}\left(\sum_{i=1}^{l} w_h\left[h_i^*(t_j) - h_i(t_j)\right]^2 + w_Q\left[\Delta Q^*(t_j) - \Delta Q(t_j)\right]^2\right)} \qquad (19)$$

where $\Delta Q^*(t_j)$ and $\Delta Q(t_j)$ are the cumulative water content changes for the whole soil column at time t_j (as computed from the forward analysis and the hmPSO algorithm respectively), $h_i^*(t_j)$ and $h_i(t_j)$ are the pressure heads at a depth of z_i and time t_j (as computed from the forward analysis and the hmPSO algorithm respectively), l is the number of depths at which the values of pressure head are computed, p is the number of times when the values of pressure head and water content are computed and w_h and w_Q are weighting factors accounting for the difference in units of the two additive terms of pressure head and water content respectively.

In the following example, the pressure heads and cumulative water content changes are computed at the six different times of 1, 2, 3, 4, 5 and 6 hours (i.e. p=6 in Eqn 19). At each of these times, pressure heads are computed at the six different depths of 0.05 0.1, 0.15, 0.2, 0.25, 0.3, 0.4, 0.5, 0.6, 0.8, and 1.0 metres (i.e. l=11 in Eqn 19). The two weighting factors w_h and w_Q are set to 1.0 and 10.0 respectively.

Table 6 shows the results from the last five NM local searches performed during a typical parallel run of the hmPSO algorithm (using 20 particles and 22 processors). The final solution corresponds to a value of objective function below the termination tolerance of 1.0E-5 with an estimated set of parameter values $\mathbf{x}$ = [3.35, 2.00, 9.22E-5, 0.368, 0.102]^T. The perfect match with the parameter values used for the forward analysis confirms the ability of the hmPSO algorithm to find the global minimum of the objective function with remarkable accuracy. The benefits of the combination of the

bPSO with the NM method are also evident from inspection of Table 6. The quality of the initial guesses by the bPSO at the simplex vertices progressively improves with the number of local searches while the increased accuracy of the NM solution focuses the search sub-domain of the local bPSO.

Table 7 shows the number of objective function evaluations performed by each of the 48 processors allocated to individual particles of the swarm. Inspection of Table 7 indicates that large differences exist in the number of objective function evaluations by different particles leading to significant imbalance between processors. This also confirms the importance of asynchronous, as opposed to synchronous, parallel implementation in order to limit the impact of variable computational speed among processors.

Table 8 shows that, as for the numerical tests, the hmPSO achieves high success rates over 25 runs (in both serial and parallel implementations) while bPSO fails to converge in any of the 25 runs. The robustness of hmPSO is also confirmed by the fact that the high success rate remains practically constant when different numbers of processors/particles are used.

It has also been noted that success rate starts to deteriorate when the number of processors exceeds 37 as the maximum allowed number of objective function evaluations is attained but the minimum of the objective function is still greater than the termination tolerance (note that the maximum number of function evaluations is fixed in Table 5 regardless of the number of processors). This happens because, as a larger number of processors is used, the overall number of function evaluations per unit time rises accordingly so, even if the run tends to last shorter, the number of function evaluations performed during the run increases. On the other hand, the number of local searches per unit time remains largely unchanged (see Figure 9) and hence a shorter computational time means a lower number of local searches. It can therefore happen that, when using more than 37 processors, a relatively low number of local searches is performed before the maximum number of function evaluations is attained. Given the key role played by local searches in the definition of the bPSO sub-domain, a reduced number of local searches might impact on the efficiency of the particle swarm and might lead to failure of the algorithm.

The convergence characteristics of the hmPSO for different runs using single and multiple processors are illustrated in Figure 10. As previously mentioned, the overall

number of function evaluations becomes significantly larger when moving from a serial single-processor implementation to a parallel implementation with 47 processors. On the other hand, Table 8 shows that computational time becomes significantly smaller when moving from a serial single-processor implementation (around 1.5 hours) to a parallel implementation with 47 processors (12 minutes). The variation of computational time and parallel speedup with number of processors is also illustrated in Figure 11.

CONCLUSIONS

The paper presents a novel “hybrid moving boundary particle swarm optimization” algorithm (hmPSO) that enables calibration of geotechnical models from laboratory or field measurements. A simulation-based optimization process is devised to match experimental data to model predictions by minimizing an objective function that measures the difference between them.

The hmPSO algorithm is the result of hybridization of a “basic particle swarm optimization” (bPSO) algorithm with a NM local search algorithm. The bPSO includes two distinct particle swarms flying over the “global” domain (i.e. exploring the entire search space) and the “local” sub-domain (i.e. exploiting the most promising search region) respectively.

Serial and parallel implementations of hmPSO are described and validated on a number of benchmark tests. These include purely numerical tests, where the minimum of multimodal mathematical functions is sought, as well as a challenging geotechnical parameter determination based on the analysis of water infiltration in an unsaturated soil column. Such application clearly demonstrates the potential of hmPSO for back-analysis of geotechnical case studies as well as for routine interpretation of laboratory and field tests.

A number of important features of hmPSO have emerged during validation of the algorithm. The combination of bPSO with the NM local search greatly improves efficiency and avoids premature convergence to a local minimum. NM local searches are also key in moving and progressively narrowing the search sub-domain exploited by the local bPSO swarm. An efficient combination of bPSO and NM local searches is implemented in this work by using a client-server model

Simulation-based hmPSO applications to geotechnical problems can be computationally very demanding and parallel implementation has been shown very effective in reducing computation time. A markedly non-linear parallel speedup has been observed during the application of hmPSO to the unsaturated flow problem considered in this work. This can be explained by considering that, as more processors are used, the number of particle function evaluations increases but this is not matched by a similar increase in the number of local searches, which would be required to maintain algorithm scalability.

Load unbalance between different processors is detrimental to the performance of parallel hmPSO but negative impact can be reduced by adopting asynchronous implementation, where each particle in the swarm is continuously updated regardless of the status of neighbouring particles.

References

1. Mattsson H, Klisinski M and Axelsson K. Optimization routine for identification of model parameters in soil plasticity. *International Journal for Numerical and Analytical Methods in Geomechanics*, 2001, 25: 435-472.
2. Zentar R, Hicher PY, and Moulin G. Identification of soil parameters by inverse analysis. *Computers and Geotechnics*, 1999, 28: 129-144.
3. Mattsson H, Axelsson K and Klisinski M. On a constitutive driver as a useful tool in soil plasticity. *Advances in Engineering Software*, 1997, 30: 661-668.
4. Kennedy J and Eberhart R. Particle swarm optimization. In *IEEE, Neural Networks Council Staff, IEEE Neural Networks Council editor Proc. IEEE International Conference on Neural Networks*, IEEE, 1995, p.1942-1948.
5. Perez R and Behdinan K. Particle Swarm Approach for Structural Design Optimization. *Computers and Structures*, 2007, 85:1579-1588.
6. Zhang W, Liu M and Clerc Y. An adaptive pso algorithm for reactive power optimization. In *Sixth international conference on advances in power system control, operation and management (APSCOM)*, Hong Kong, China, 2003, p. 302-307.
7. Fourie P and Groenwold A. The particle swarm optimization algorithm in size and shape optimization. *Structural and Multidisciplinary Optimization*, 2002, 23(4):259-267.
8. Venter G and Sobieszczanski-Sobieski J. Multidisciplinary optimization of a transport aircraft wing using particle swarm optimization. *Structural and Multidisciplinary Optimization*, 2004, 26(1):121-131.
9. Weise T. Global Optimization Algorithms - Theory and Application. *Free ebook at www.it-weise.de/projects/book.pdf*.

10. Xie X, Zhang W and Yang Z. A dissipative particle swarm optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02)*, Hawaii, USA, 2002, p. 1456–1461.
11. Dos Santos Coelho L and Mariani V. Particle swarm optimization with quasi-Newton local search for solving economic dispatch problem. In *IEEE International Conference on Systems, Man and Cybernetics*, Taipei, Taiwan, Piscataway, NJ, IEEE Press, 2006, p. 3109-3113.
12. Nelder J and Mead R. A simplex method for function minimization. *The Computer Journal*, 1965, 7:308–313.
13. Renders J and Flasse S. Hybrid methods using genetic algorithms for global optimization. *IEEE Trans Syst Man Cybern B Cybern*,1996, 26(2):243–258.
14. Yen R, Liao J, Lee B and Randolph D. A hybrid approach to modeling metabolic systems using a genetic algorithm and Simplex method. *IEEE Transactions on Systems, Man and Cybernetics Part-B*, 1998, 28(2):173–191.
15. Fan S, Liang Y and Zahara E. Hybrid simplex search and particle swarm optimization for the global optimization of multimodal functions. *Engineering Optimization*, 2004, 36:401–418.
16. Schutte J, Reinbolt J, Fregly B, Haftka R and George A. Parallel global optimization with the particle swarm algorithm. *International Journal of Numerical Methods in Engineering*, 2004, 61(13):2296-2315.
17. Eberhart R and Kennedy J. A new optimizer using particle swarm theory. In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, Nagoya, Japan, 1995, p. 39–43.
18. Shi Y and Eberhart R. Parameter selection in particle swarm optimization. In *Proceedings of the 1998 Annual Conference on Evolutionary Computation*, Springer-Verlag, New York, 1988, p. 591-600.
19. Shi Y and Eberhart R. A modified particle swarm optimizer. In *IEEE International Conference on Evolutionary Computation*, Anchorage, Alaska, 1998, p. 69-73.
20. Shi Y and Eberhart R. Parameter selection in particle swarm optimization. In *Proceedings of the Seventh Annual Conference on Evolutionary*, New York, Springer,1998, p. 591–600.
21. Hu X and Eberhart R. Adaptive particle swarm optimization: detection and response to dynamic systems. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002)*, Honolulu, Hawaii, USA, 2002, p.1666-1670.
22. Clerc M. Tribes - a parameter free particle swarm optimizer. *http://www.mauriceclerc.net*.
23. Løbjerg M, Rasmussen T and Krink K. Hybrid particle swarm optimizer with breeding and subpopulations. In *Proc. of the Third Genetic and Evolutionary Computation Conference*, 2001, 1:469-476.
24. Løvbjerg M and Krink T. Extending particle swarm opimisers with self-organized criticality. In *Proceedings of Fourth Congress on Evolutionary Conference*, 2002, p. 1588-1593.
25. van den Bergh F. An analysis of particle swarm optimizers. *PhD Thesis, Department of Computer Science, University of Pretoria*, Pretoria, South Africa, 2002.

26. Bird S and Li X. Adaptively choosing niching parameters in a PSO. In *Proceedings of the 8th Conference on Genetic and Evolutionary Computation*, Seattle, Washington, USA, 2006, p.3-10.

27. Brits R, Engelbrecht A and Bergh B. A niching particle swarm optimizer. In *Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02)*, Orchid Country Club, Singapore, 2002, p.692--696.

28. Vesterstrom J, Riget J and Krink T. Division of labor in particle swarm optimisation. In *IEEE 2002 Proceedings of the Congress on Evolutionary Computation*, 2002, p. 1570-1575

29. Yun Y. Hybrid genetic algorithm with adaptive local search scheme. *Computers and Industrial Engineering*, 2006; 51(1): 128-141.

30. Das S, Koduru P, Gui M, Cochran M, Wareing A, Welch S and Babin B. Adding local search to particle swarm optimization. In *Proc. IEEE 2006 Congress on Evolutionary Computation (CEC 2006)*, Vancouver, Canada, 2006, p. 428-433.

31. Fan S and Zahara E. A hybrid simplex search and particle swarm optimization for unconstrained optimization. *European Journal of Operational Research*, 2007, 181(2):527-548.

32. Genovesi S, Monorchio A, Mittra R and Manara G. A sub-boundary approach for enhanced particle swarm optimization and its application to the design of artificial magnetic conductors. *IEEE Transactions on Antennas and Propagation*, 2007; 55(3):766-770.

33. Cantu-Paz E. Markov chain models of parallel genetic algorithms. *IEEE Transactions on Evolutionary Computation*, 2000, 4(3):216-226.

34. Suganthan P, Hansen N, Liang J, Deb K, Chen Y, Auger A and Tiwari S. Problem definitions and evaluation criteria for the CEC 2005 special session on real parameter optimization. National Chiao Tung University, Natural Computing Laboratory, Hsinchu, Taiwan, *NC Lab Report No. NCL-TR-2005001*, 2005.

35. Snir M, Otto S, Huss-Lederman S, Walker D and Dongarra J. MPI: The Complete Reference. MIT Press, 1996.

36. Van Genuchten M. A closed form equation for predicting the hydraulic conductivity of unsaturated soils. *Soil Sci. Soc. Am. J.*, 1980, 44:892-898.

37. Mualem Y. A new model predicting the hydraulic conductivity of unsaturated porous media. *Water Resour. Res.,* 1976, 12:513–522.

38. Zhang Y, Augarde C, Gallipoli D. Identification of hydraulic parameters for unsaturated soils using particle swarm optimization. In *Proceedings 1st European Conference on Unsaturated Soils*, Durham, United Kingdom, 2008, p.765-771

39. Celia M, Bouloutas E and Zarba R. A general mass-conservative numerical solution for the unsaturated flow problem. *Water Resources Research*, 1990, 26(7):1483-1496.

40. Simunek J, van Genuchten M, Gribb M and Hopmans J. Parameter estimation of unsaturated soil hydraulic properties from transient flow processes. *Soil & Tillage Research*, 1998, 47(1):27-36.

TABLES

Table 1: Mathematical functions for numerical tests

Function $f(\mathbf{x})$	**Features**	**Dimension**	**Range**	**Global minimum**
Rasgrigin	Multimodal	n=5, n=10	$[-5.12, 5.12]^n$	$g(\mathbf{x}^*)=0$
Griewank	Multimodal, shifted	n=5, n=10	$[-600, 600]^n$	$g(\mathbf{x}^*)=0$
Ackley	Multimodal, shifted	n=5, n=10	$[-32, 32]^n$	$g(\mathbf{x}^*)=-140$
Sphere	Unimodal, shifted	n=5, n=10	$[-100, 100]^n$	$g(\mathbf{x}^*)=-450$
Rosenbrock	Multimodal, shifted	n=5, n=10	$[-100, 100]^n$	$g(\mathbf{x}^*)=390$

Table 2: Settings for hmPSO in numerical tests

Parameter	**Value**
Swarm size	20 particles for n=5 40 particles for n=10
Range of inertia weight, $[w_{\min}, w_{\max}]$	[0.4, 0.75]
Cognitive weight, c_1	2.0
Social weight, c_2	2.0
Maximum velocity, $\mathbf{v}_{\max}$	0.3 $(\mathbf{x}_u - \mathbf{x}_l)$
Termination tolerance	1.0 e-5
Maximum number of function evaluations	10000 n
Range of sub-domain radius, $[\delta_{\min}, \delta_{\max}]$	[0.2, 0.4]
Maximum iteration number of Nelder-Mead method	300

Table 3: Success rates during numerical tests

Function	Algorithm	Success rate
Shifted Sphere	hmPSO	100%
n=5	bPSO	28%
Shifted Sphere	hmPSO	100%
n=10	bPSO	8%
Shifted Rosenbrock	hmPSO	100%
n=5	bPSO	0%
Shifted Rosenbrock	hmPSO	72%
n=10	bPSO	0%
Rasgrigin	hmPSO	96%
n=5	bPSO	96%
Rasgrigin	hmPSO	12%
n=10	bPSO	16%
Shifted Griewank	hmPSO	8%
n=5	bPSO	4%
Shifted Griewank	hmPSO	96%
n=10	bPSO	0%
Shifted Ackley	hmPSO	100%
n=5	bPSO	64%
Shifted Ackley	hmPSO	100%
n=10	bPSO	12%

Table 4. Parameter ranges for the Mualem-van Genuchten model [33]

Parameter	Minimum value	Maximum value
β	1.001	3.5
α	0.1 m^{-1}	9.6 m^{-1}
θ_s	0.21	0.7
θ_r	0.001	0.2
K_s	5.0E-8 m/s	5.0E-4 m/s

Table 5: Settings for hmPSO in simulation-based optimization test

Parameter	**Value**
Swarm size	30 particles per processor
Range of inertia weight, $[w_{\min}, w_{\max}]$	[0.4, 0.75]
Cognitive weight, c_1	2.0
Social weight, c_2	2.0
Maximum velocity, $\mathbf{v}_{\max}$	0.3 ($\mathbf{x}_u - \mathbf{x}_l$)
Termination tolerance	1.0 e-5
Maximum number of evaluations of the objective function	500000
Range of sub-domain radius, $[\delta_{\min}, \delta_{\max}]$	[0.2, 0.4]
Maximum number of NM transformations	300

Table 6: Sequence of NM local searches in parallel hmPSO for a run using 20 particles

Local search counter		Objective function value	Parameters				
			β	α	θ_r	θ_s	K_s
1	initial guesses at vertices of simplex	1.59E+00	2.945	1.927	0.081	0.529	9.39E-05
		1.86E+00	4.096	1.994	0.210	0.347	9.54E-05
		2.51E+00	2.483	2.389	0.171	0.612	8.99E-05
		3.19E+00	4.511	2.457	0.210	0.355	3.61E-04
		3.94E+00	3.537	1.807	0.136	0.578	1.22E-04
		4.02E+00	1.684	2.634	0.051	0.554	3.34E-05
	solution	6.08E-02	2.924	2.016	0.083	0.526	1.00E-04
2	initial guesses at vertices of simplex	6.08E-02	2.924	2.016	0.083	0.526	1.00E-04
		2.61E-01	3.561	1.961	0.087	0.395	1.23E-04
		4.79E-01	3.274	1.956	0.064	0.604	1.62E-04
		5.06E-01	4.689	1.910	0.095	0.476	3.41E-04
		5.23E-01	3.414	1.943	0.104	0.408	1.03E-04
		5.27E-01	3.948	1.911	0.050	0.851	4.07E-04
	solution	3.60E-02	3.014	2.012	0.092	0.450	8.91E-05
3	initial guesses at vertices of simplex	3.60E-02	3.014	2.012	0.092	0.450	8.91E-05
		1.44E-01	3.414	1.976	0.101	0.382	9.95E-05
		2.22E-01	3.617	1.961	0.072	0.646	2.40E-04
		2.61E-01	3.561	1.961	0.087	0.395	1.23E-04
		4.79E-01	3.274	1.956	0.064	0.604	1.62E-04
		5.06E-01	4.689	1.910	0.095	0.476	3.41E-04
	solution	1.54E-02	3.178	2.005	0.098	0.401	8.90E-05
4	initial guesses at vertices of simplex	1.54E-02	3.178	2.005	0.098	0.401	8.90E-05
		1.42E-01	3.160	2.001	0.090	0.498	1.16E-04
		1.44E-01	3.414	1.976	0.101	0.382	9.95E-05
		1.63E-01	3.166	2.001	0.071	0.497	1.23E-04
		1.67E-01	3.450	2.007	0.106	0.381	1.06E-04
		2.22E-01	3.617	1.961	0.072	0.646	2.40E-04
	solution	7.32E-04	3.345	2.000	0.102	0.370	9.26E-05
5	initial guesses at vertices of simplex	7.42E-04	3.345	2.000	0.102	0.370	9.26E-05
		6.90E-02	3.602	1.993	0.079	0.657	2.53E-04
		8.91E-02	3.629	1.988	0.109	0.343	1.04E-04
		9.65E-02	3.883	1.988	0.079	0.590	2.85E-04
		1.42E-01	3.160	2.001	0.090	0.498	1.16E-04
		1.44E-01	3.414	1.976	0.101	0.382	9.95E-05
	solution	2.73E-06	3.350	2.000	0.102	0.368	9.22E-05

Table 7: Number of evaluations of the objective function for different particles allocated to different processors (for a specific run with 48 particles and 50 processors)

Particle no.	Function evaluations	Particle no.	Function evaluations	Particle no.	Function evaluations
1	7211	17	8383	33	8016
2	9268	18	8049	34	6850
3	8274	19	7164	35	7028
4	6989	20	6994	36	6706
5	6417	21	6555	37	7177
6	7196	22	6845	38	6733
7	6754	23	6976	39	6495
8	5757	24	7140	40	7339
9	8926	25	7505	41	6546
10	9052	26	8139	42	7290
11	8095	27	6988	43	6017
12	7235	28	7117	44	7248
13	7287	29	7172	45	6229
14	6330	30	7300	46	6813
15	6286	31	7284	47	7327
16	6982	32	6065	48	7825

Table 8: Efficiency and success rate over 25 runs for bPSO, hmPSO (serial) and hmPSO (parallel)

	No. of particles	No. of processors	Computational time (s)			Parallel speedup	Success rate (%)
			Lowest	Largest	Mean		
bPSO	30	1	47921	77638	69087		0
hmPSO (serial)	30	1	1352	46666	5595		100
hmPSO (parallel)	15	17	530	2278	1147	4.87	96
	20	22	349	4624	1176	4.75	100
	30	32	454	1510	903	6.19	100
	35	37	412	1292	823	6.79	100
	45	47	349	1093	719	7.77	96

FIGURES

```
foreach particle i = 1, . . . , Np do
        xi      ← generate random xi ∈[xl , xu];
        vi      ← generate random vi ∈[-vmax , +vmax];
        f(xi)   ← evaluate objective function ;
        if (f(xi) < f(Pi))          Pi← xi
        if (f(xi) < f(Pg))          Pg← xi
end foreach

k←0
IL←0
repeat
        foreach particle i = 1, . . . , Np do
                update velocity vi using Eq. 2
                apply bounds constraints on vi
                update position xi
                apply bounds constraints on xi
                f(xi)   ← evaluate objective function;
                if (f(xi) < f(Pi))          Pi← xi
                if (f(xi) < f(Pg))          Pg← xi
        end foreach particle

        if (mod (k, NL) =0)
                sortParticles( )
                xL = doLocalSearch(best n+1 particles) ;
                restartSubSwarm (particleIDs), [xL-δI , xL+δI] ;
                IL← IL + 1
        endif
        k← k+1
while (stop criteria is not met)
```

Fig. 1: Serial hmPSO algorithm

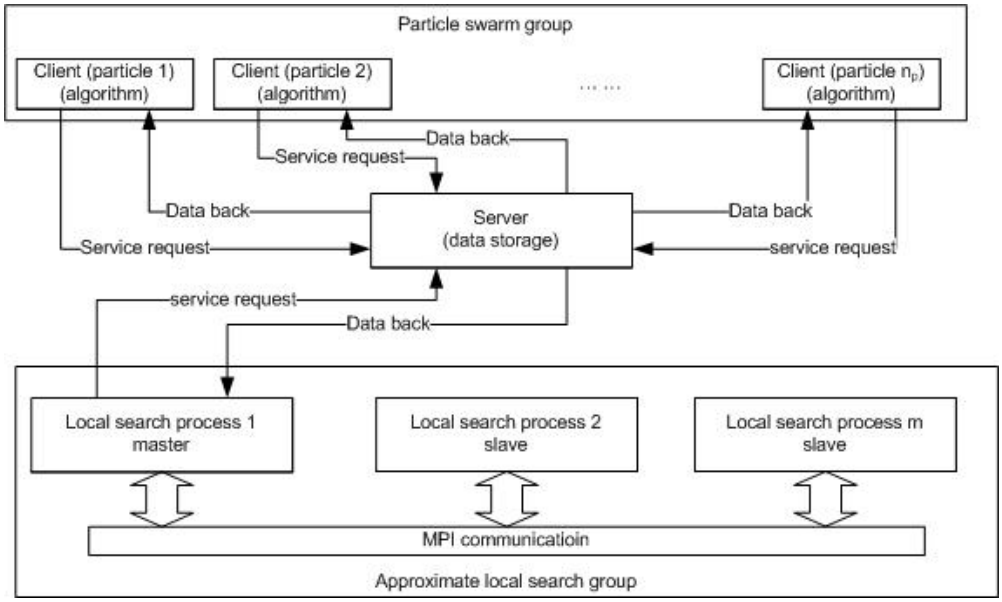

Fig. 2: Client-server model for parallel hmPSO

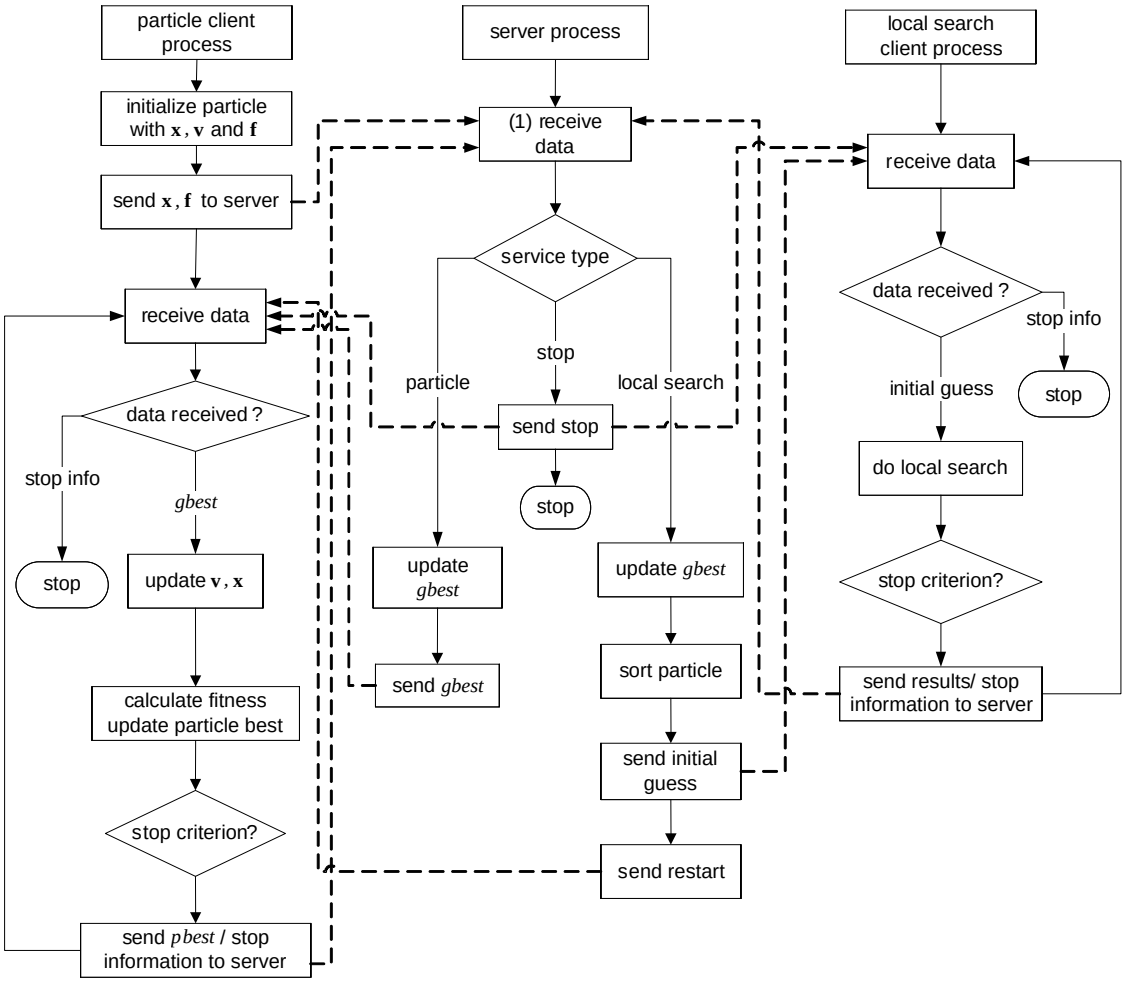

Fig. 3. Flowchart of the parallel hmPSO

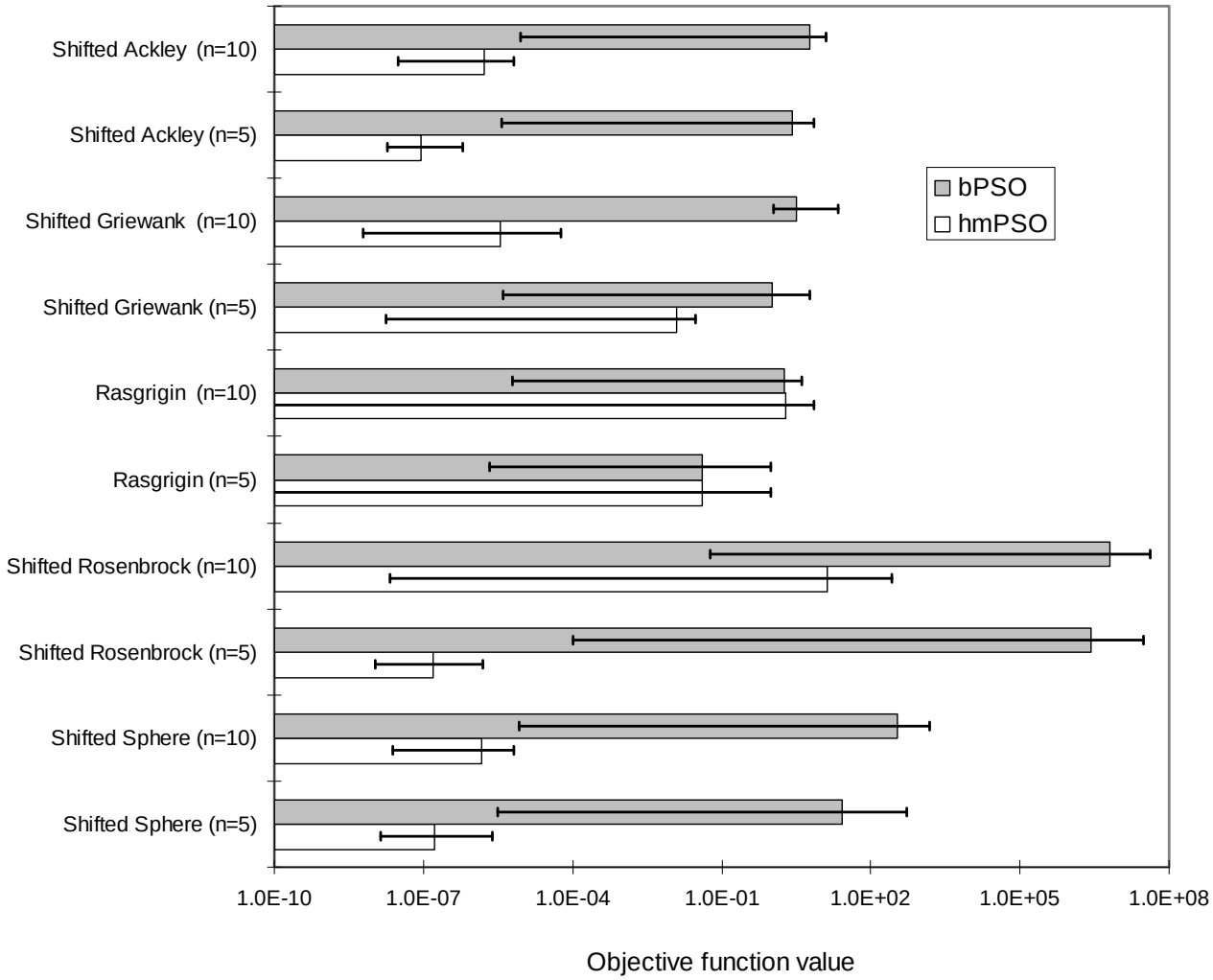

Fig. 4. Objective function values during numerical tests

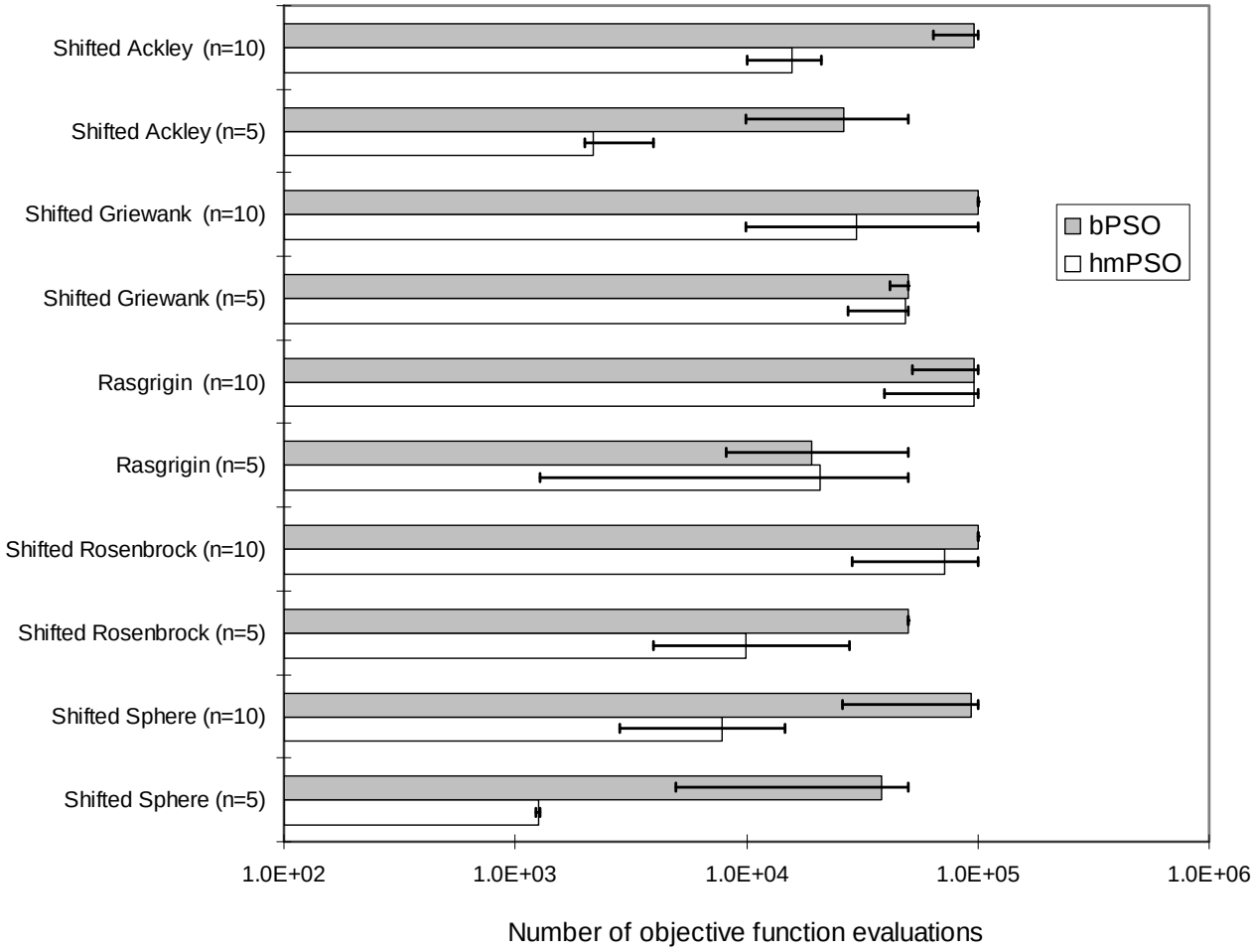

Fig. 5. Number of objective function evaluations during numerical tests

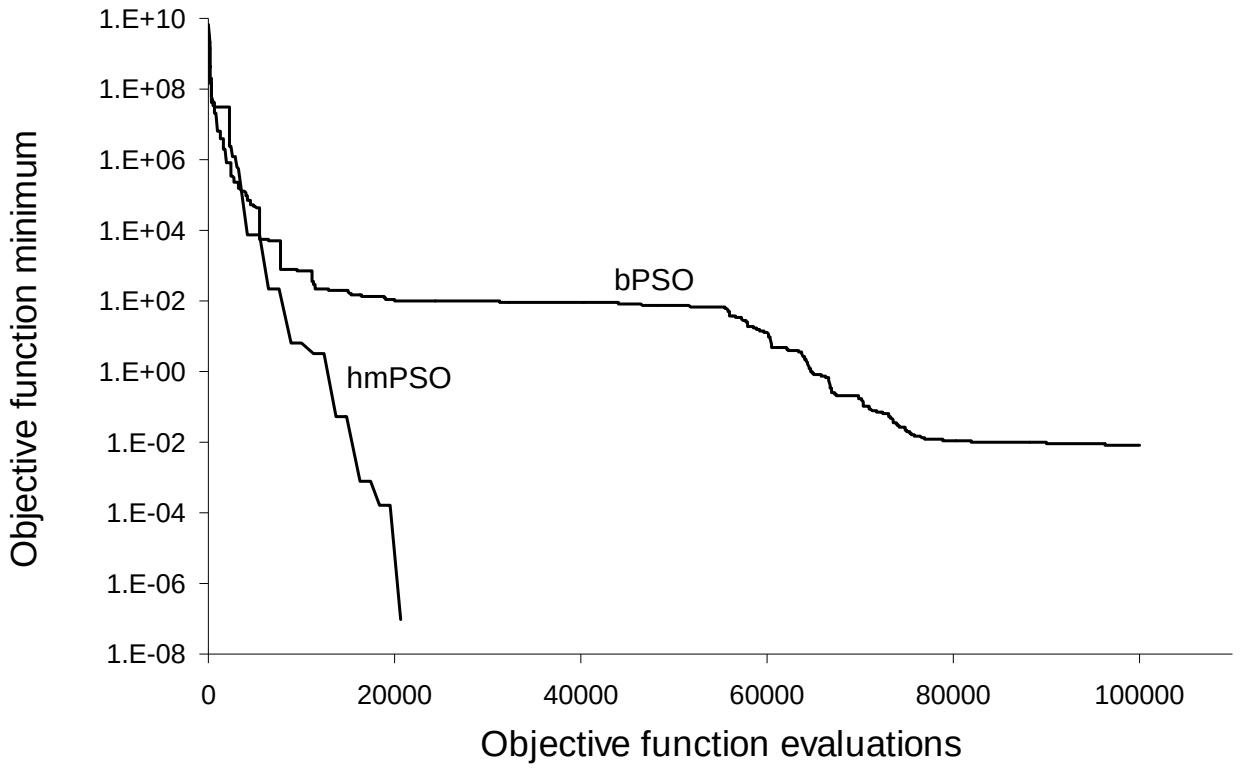

Fig. 6: Convergence characteristics of bPSO and hmPSO for Shifted Rosenbrock (n=10) function

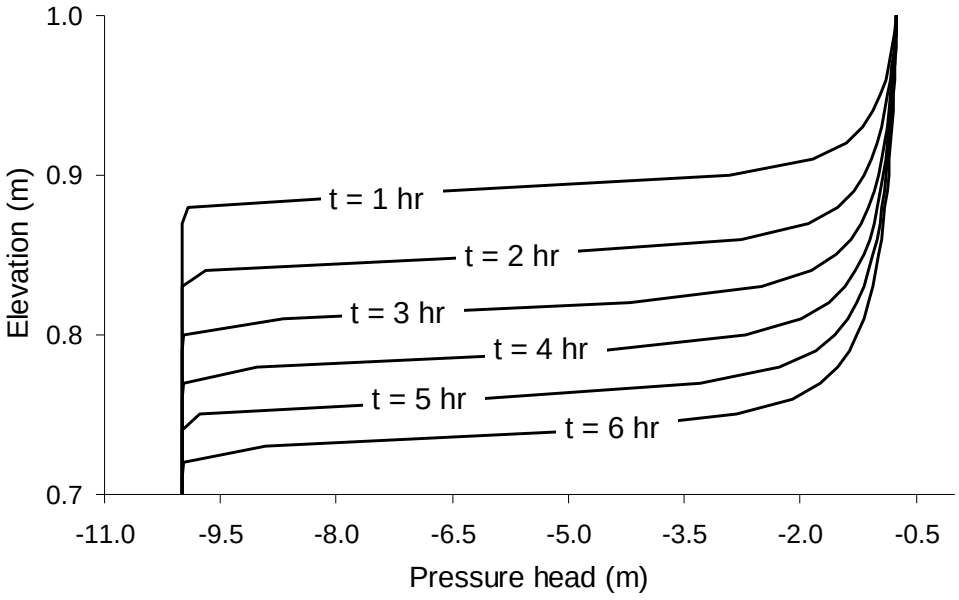

Figure 7. Water pressure head profile.

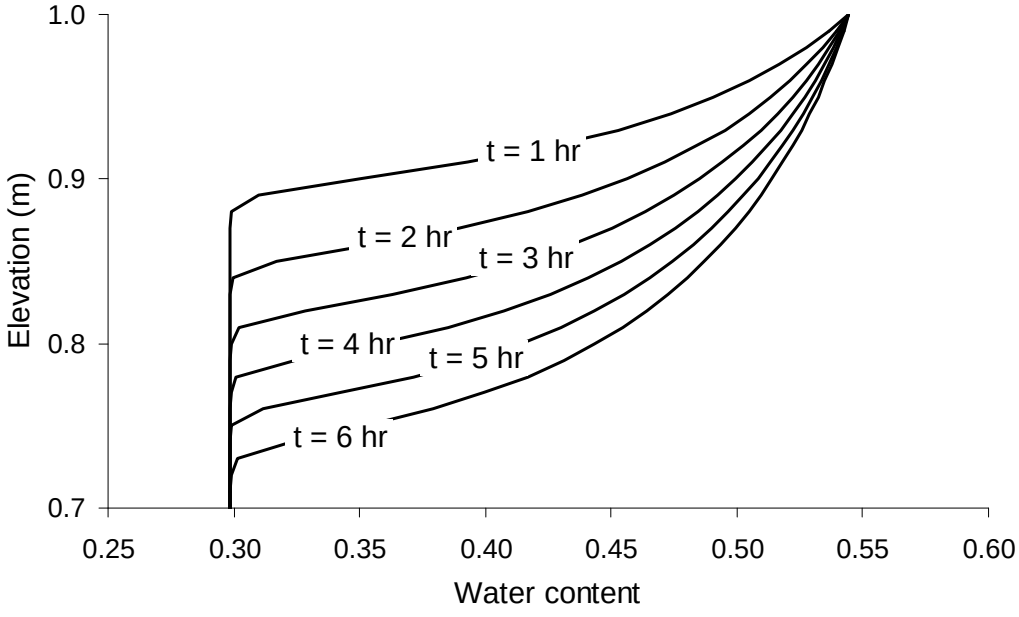

Figure 8. Water content profile

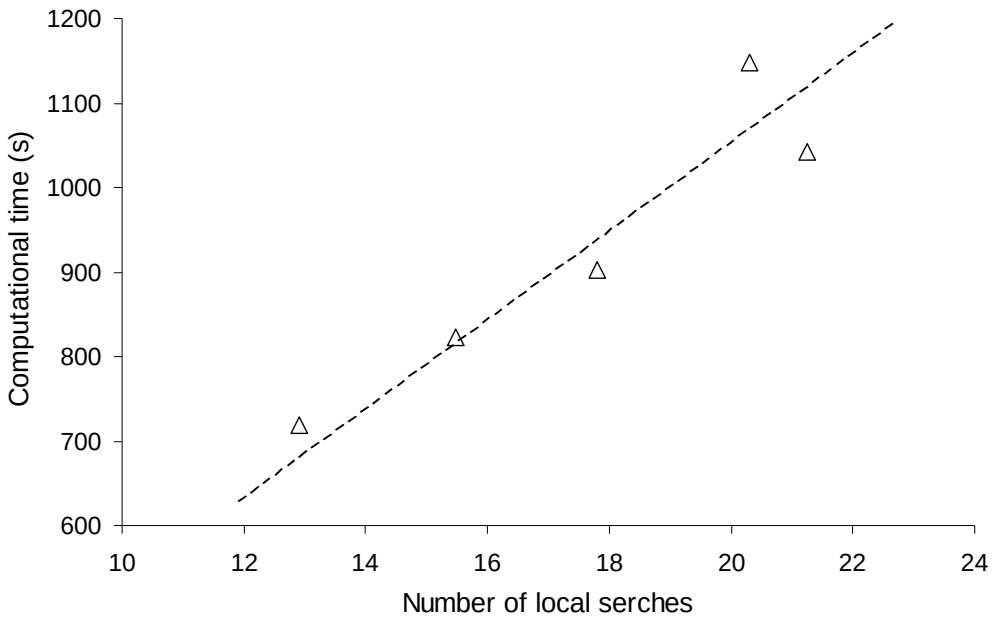

Figure 9. Number of local searches versus computational time for parallel hmPSO. Each point represents the average over successful runs out of a total of 25 runs when using 17, 22, 32, 37 and 47 processors.

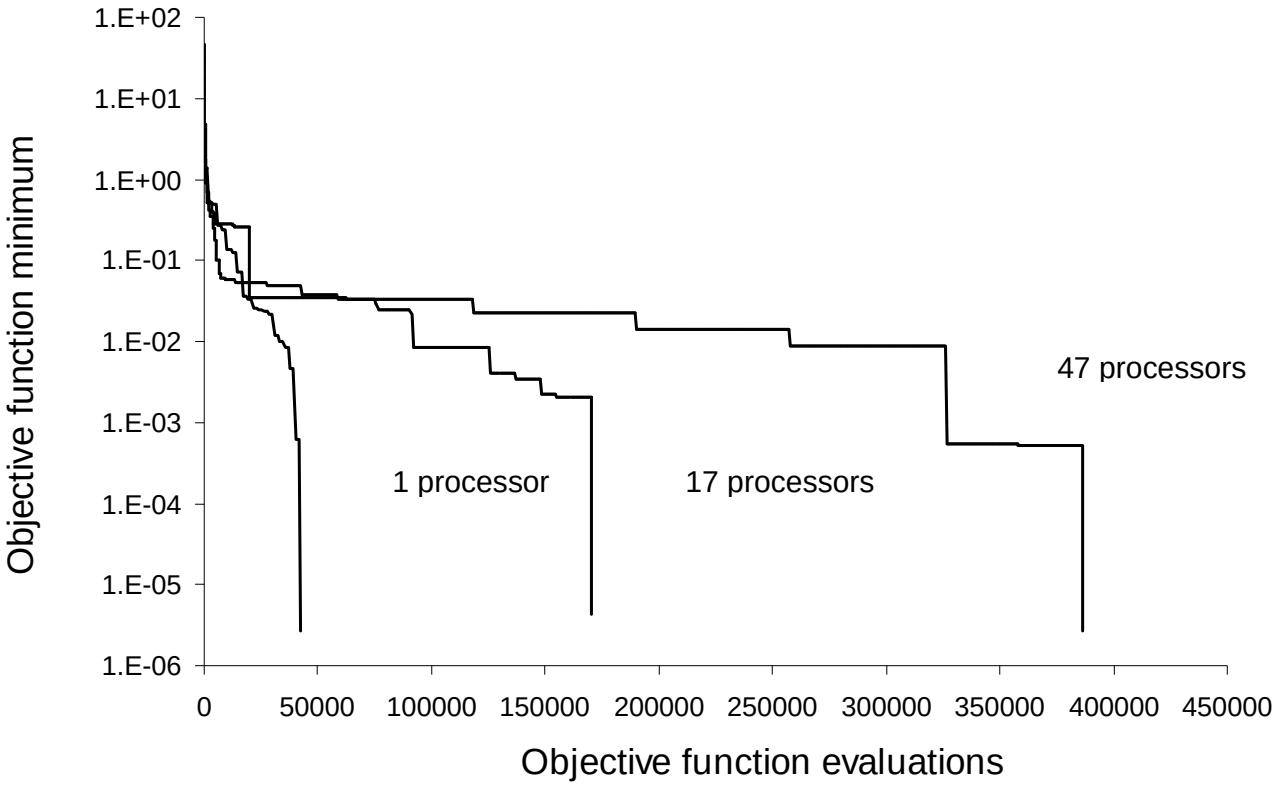

Fig. 10: Convergence characteristics for serial and parallel hmPSO

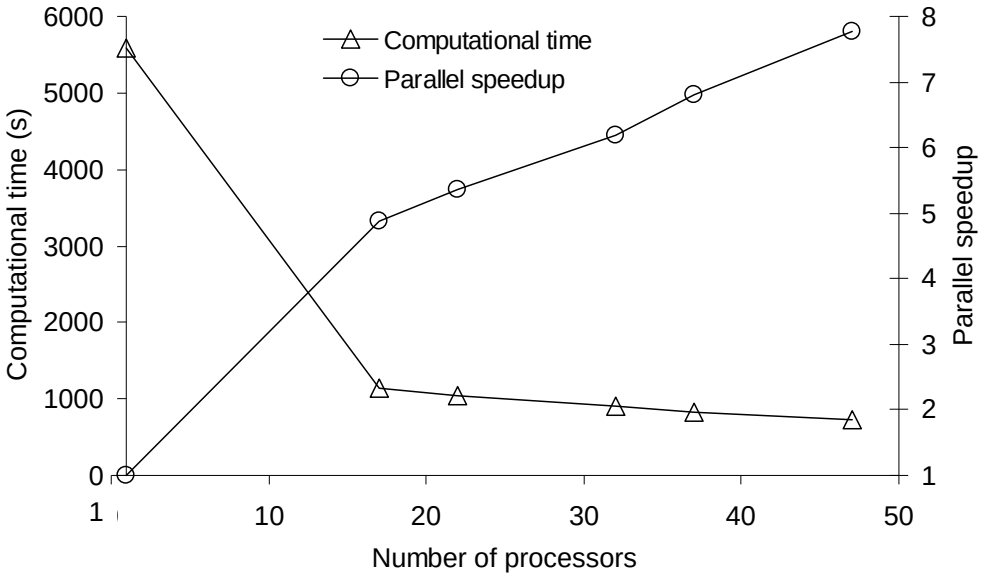

Figure 11. Computational time and parallel speedup versus number of processors for hmPSO. Each point represents the average over successful runs out of a total of 25 runs when using 1, 17, 22, 32, 37 and 47 processors.