COMPARISON OF MODERN HEURISTICS ON SOLVING THE PHASE STABILITY TESTING PROBLEM

Tomáš Smejkal*, Jiří Mikyška and Jaromír Kukal

Czech Technical University in Prague
Faculty of Nuclear Sciences and Physical Engineering
Trojanova 13, 120 00 Prague 2, Czech Republic

Abstract. In this paper, we are concerned with the phase stability testing at constant volume, temperature, and moles (VTN-specification) of a multicomponent mixture, which is an unconstrained minimization problem. We present and compare the performance of five chosen optimization algorithms: Differential Evolution, Cuckoo Search, Harmony Search, CMA-ES, and Elephant Herding Optimization. For the comparison of the evolution strategies, we use the Wilcoxon signed-rank test. In addition, we compare the evolution strategies with the classical Newton-Raphson method based on the computation times. Moreover, we present the expanded mirroring technique, which mirrors the computed solution into a given simplex.

1. Introduction. The phase stability testing of multicomponent mixtures is a basic problem in chemical engineering with multiple applications in the industry. Development of fast and robust algorithms to solve this problem is important, e.g., for hydrocarbon reservoirs simulation or in CO$_2$ sequestration.

In the phase stability testing, we are interested whether the given mixture is stable or the splitting in two or more phases occurs. Solving the phase stability problem results into finding the global minima of certain functions called Tangent Plane Distance (TPD). The classical approach [8, 11, 25] to find this minimum is using the Newton-Raphson method. As the Newton-Raphson method is only local, more than one initial approximation of the solution has to be used to find the global minimum [13]. The number of initial approximations can depend on the number of components and, therefore, for mixtures with many components, the calculation time can exceed our demands. This unpleasant property led us to investigate alternative algorithms, which will be better than the Newton-Raphson method. In this paper, we will investigate five heuristics: the Differential Evolution (DE), the Cuckoo Search (CS), the Harmony Search (HS), the Covariance Matrix Adaptation (CMA-ES), and the Elephant Herding Optimization (EHO).

The Differential Evolution was first presented by Storm and Price [28]. Since then, many variants of the Differential Evolution have been published [3, 21, 43]. In recent years the main focus is on the adaptability of the parameters. The Cuckoo Search is an evolution algorithm introduced by Yang [40] and is motivated by the

2010 Mathematics Subject Classification. 49M15, 90C26, 90C59, 65K10, 49M37.

Key words and phrases. Optimization, phase stability testing, Newton-Raphson method, VTN-specification, Differential Evolution, Cuckoo Search, Harmony Search, Covariance Matrix Adaptation, elephant herding optimization.

* Corresponding author: Tomáš Smejkal.
brood parasitic behavior of cuckoos. The main advantage of the Cuckoo Search is its simplicity and low number of parameters. Therefore, parameter adjustment is not as difficult as in the Differential Evolution. The Harmony Search algorithm was created by Geem [5]. It is a heuristic algorithm inspired by the improvisation process of jazz musicians. Compared to other heuristics, the Harmony Search is the simplest to implement, however, the method has more parameters to adjust. The fourth chosen algorithm is the CMA-ES algorithm developed by Hansen [6, 15]. The method is based on generating the population from the normal distribution. The mean and covariance matrix of the normal distribution are calculated from the best \( \mu \) members of the population. The last heuristic, the Elephant Herding Optimization (EHO), was created by Wang [30]. The algorithm is based on the behaviour of the elephants in nature, where the herd of elephants is led by a matriarch, and the male elephants leave the herd. The detailed explanation of all algorithms is presented in Section 3.

In Table 1, we summarise the main advantages and disadvantages of the evolution strategies.

In addition to the presented algorithms, there exist many other evolution algorithms. In recent years, the research focused on the meta-heuristic algorithms [29]. Noteworthy are Krill Herd (KH) algorithms [31, 32, 33, 34, 35, 36], non-dominated sorting genetic algorithms [41, 42], or algorithms based on Hurricane Optimization [23].

In this paper, we are interested in two main aspects of the presented methods: accuracy and speed. If the phase stability test wrongly predicts stability, the solution of the phase equilibrium calculation will fail because the classical phase equilibrium algorithms [8, 12, 25] are terminated when the phase stability test indicates a stable mixture. The second observed aspect is the speed of the computation. As previously stated, the methods have to find the solution in a reasonable time because the stability testing can be used in, e.g., compositional simulation [19, 37, 38, 39], where the stability test is performed on each finite element of the discretized domain at each time step. Therefore, a single criterion, which combines these two requirements, is adopted from [1]. Then, the Wilcoxon signed-rank test is performed to compare the individual algorithms.

The structure of this paper is as follows. In Section 2, the phase stability testing problem at constant volume, temperature, and moles (known as \( VTN \)-specification) will be discussed, and the objective function for the optimization problem will be defined. In Section 3, the detailed description of the Newton-Raphson method and the evolution algorithms will be provided. In Section 4, the comparison of the numerical solutions of chosen problems will be given. In Section 5, we summarize results and draw some conclusions.

2. Phase stability testing at constant volume, temperature and moles. In the single phase stability testing problem, the task is to find out whether the single phase is stable or the system will split into two (or more) phases. The classical approach to solve this problem is to test whether a state can be found which, if taken in a small amount from the initial phase, will lead to a better value of the appropriate thermodynamical potential. In the case of the phase stability testing at constant volume, temperature, and moles (i.e., \( VTN \)-specification), the appropriate thermodynamical potential is the Helmholtz free energy density \( a \), which is defined as

\[
a = \sum_{i=1}^{n} \mu_i c_i - P,
\]
The advantages and disadvantages of the chosen evolution algorithms.

|       | advantages                                      | disadvantages                                      |
|-------|------------------------------------------------|---------------------------------------------------|
| DE    | • good convergence properties                  | • parameter tuning is necessary                    |
|       | • strong theoretical analysis                   | • easy to stuck in a local minimum                 |
| CS    | • supports local and global search              | • small precision                                 |
|       | • easy to hybridize                             | • no theoretical analysis                          |
| HS    | • simple implementation                         | • slow convergence                                 |
|       | • small population                              | • small precision                                 |
| CMA-ES| • no curse of dimensionality [2]                | • harder implementation                           |
|       | • in-variance properties                        | • a lot of parameters                              |
| EHO   | • hard to stuck in a local minimum              | • slow convergence                                 |
|       | • fewer parameters                              | • fixed parameters                                 |

Table 1. The advantages and disadvantages of the chosen evolution algorithms.

where $c_i$ is the concentration of the $i$-th component, $\mu_i$ is the chemical potential of the $i$-th component, $n$ is the number of components, and $P$ is pressure. Details about these functions are provided in the Appendix. According to [13], mixture with initial concentrations $x^* = (c_1^*, \ldots, c_n^*)^T \in D \subset \mathbb{R}^n$ is stable if and only if function $TPD = TPD(c_1, \ldots, c_n; x^*)$ defined by

$$TPD = \sum_{i=1}^{n} c_i [\mu_i (c_1, \ldots, c_n) - \mu_i (x^*)] - [P (c_1, \ldots, c_n) - P (x^*)]$$  \hspace{1cm} (2)

is non-negative for any feasible concentration $y = (c_1, \ldots, c_n)^T \in D$, where $D$ is the feasible domain, which is presented in the Appendix. On the other hand, if there exists a state $y \in D$ for which $TPD(y; x^*) < 0$, then the mixture is unstable and the splitting will occur.

3. Numerical algorithms for the single phase stability testing. To find out whether for a given $x^* \in D$, there exists a state $y \in D$ for which $TPD(y; x^*) < 0$, we will seek the global minimum of $TPD$ in the feasible domain $D$ [11, 13]. In this section, we will present existing methods, which we chose for the comparison. First, the Newton-Raphson method, which is a deterministic gradient method, will be discussed. Second, the chosen heuristics will be presented. All presented heuristics are evolution algorithms which iterate a population $\mathcal{P} = \{x^{(i)}; i \in \hat{NP}\} \subset D$ until a stopping criterion is met. The symbol $\hat{NP}$ denotes the set of positive integers not exceeding $NP$, i.e., $1, \ldots, NP$. In all presented heuristics, the initial population is generated randomly inside the feasible domain $D$. In the case of the Newton-Raphson method, we are using up to 6 initial approximations, and the strategy is taken over from [13].

3.1. Classical Newton-Raphson method. In the Newton-Raphson method, the solution is updated as

$$y^{(k+1)} = y^{(k)} + \lambda \Delta y^{(k)},$$  \hspace{1cm} (3)

where the solution increment $\Delta y^{(k)}$ is found as a solution of a system of linear equations

$$\mathbf{H} \Delta y^{(k)} = -\mathbf{g},$$  \hspace{1cm} (4)
where $\mathbb{H}$ is the Hessian matrix and $\mathbf{g}$ is the gradient of the objective function (2). Details about the form of the Hessian matrix and the gradient can be found in [13]. In equation (3), $\lambda \in (0,1]$ is damping factor which prevents overshooting. First, it is set to $\lambda = 1$ and tested whether $y^{(k+1)}$ is in the feasible domain $\mathcal{D}$. If not, the damping factor is halved, until the solution $y^{(k+1)}$ is in the feasible domain $\mathcal{D}$. The system (4) is solved using the modified Cholesky decomposition [24]. The use of the modified Cholesky decomposition ensures that the value of the objective function decreases in each iteration, therefore, the convergence toward a saddle point or to a maximum is avoided.

3.2. Differential Evolution. The DE algorithm consists of three steps: mutation, crossover and selection. In the mutation step, a parent $\mathbf{x} \in \mathcal{P}$ is chosen from the current population and then a new mutant $\mathbf{y}$ is created. In our computations, we are using version $DE/RAND/2$ [20] where the mutation operator has the form

$$y_j = x_j^{(r_1)} + F \left( x_j^{(r_2)} - x_j^{(r_3)} \right) + F \left( x_j^{(r_4)} - x_j^{(r_5)} \right),$$

(5)

where $r_i \sim U_d(1,NP)$ are independent and identically distributed randomly generated integers from 1 to $NP$ and $F$ is a scaling parameter. In the literature, there are many suggestions on how to choose parameter $F$. The optimum range of $F$ is usually between 0.4 and 1 [20]. In [28], authors suggested $F = 0.5$. In recent years, authors considered adaptation of the parameter during calculation. In SaDE [21] variation of the Differential Evolution, the parameter $F$ is approximated by a normal distribution with mean value 0.5 and standard deviation 0.3. In our calculations, we are changing $F$ in each generation using the formula

$$F = F_{\text{max}} \exp \left\{ \ln \frac{F_{\text{min}}}{F_{\text{max}}} \frac{it}{\text{maxit}} \right\} = F_{\text{max}} \left( \frac{F_{\text{min}}}{F_{\text{max}}} \right)^{\frac{it}{\text{maxit}}},$$

(6)

where $F_{\text{max}} = 1.0$, $F_{\text{min}} = 0.1$, $it$ is the current iteration index and $\text{maxit}$ is the maximum number of iterations. The main idea behind formula (6) is that at the beginning, the coefficient $F$ is close to 1 and we are exploring the whole search space. As generations progress, parameter $F$ is getting smaller and we expect faster convergence. In the crossover step, the mutant vector is crossbred with its parent $\mathbf{x}$ to create a crossover child $\mathbf{z}$ by formula

$$z_j = \begin{cases} y_j, & \eta \leq CR \text{ or } j = j_{\text{rand}}, \\ x_j, & \text{otherwise}, \end{cases}$$

(7)

where $j_{\text{rand}} \sim U_d(1,n)$ is one random integer chosen for all $j \in \hat{n}$, $\eta \sim U_c(0,1)$ is independent and identically distributed randomly generated real number between 0 and 1, and $CR$ is the crossover rate parameter. Similarly as in the case of $F$, there exist many suggestions on how to choose this parameter. We are using a strategy proposed in [14], where the parameter $CR$ is calculated using formula

$$CR = CR_{\text{max}} + \left( CR_{\text{min}} - CR_{\text{max}} \right) \left( 1.0 - \frac{it}{\text{maxit}} \right)^k,$$

(8)

In our calculation we are using $CR_{\text{max}} = 0.9$, $CR_{\text{min}} = 0.1$ and $k = 4$. In the last step, selection, the crossover child $\mathbf{z}$ replaces its parent in the population if its value of the objective function is lower. These three steps are performed for each parent $\mathbf{x} \in \mathcal{P}$ until a stopping criteria is met. In our computation, the population size is
set to $NP = 20$. A pseudo code of the Differential Evolution algorithm is presented in Algorithm 1.

### 3.3. Cuckoo Search.

In contrast to the Differential Evolution, the new member of the population in the Cuckoo Search is created using the Levy flight. The Levy flight is a process driven by an $\alpha$-stable distribution and it was shown [16, 17, 22] that this process has similar behavior as a flight of many animals and insects. The $n$-dimensional $\alpha$-stable distribution $L(\alpha, n)$ has probability density function (PDF)

$$g(x, \alpha, n) = \frac{1}{(2\pi)^{n/2} F_n} \exp\left(-\|\omega\|^\alpha\right),$$  

where $\alpha \in (0, 2)$, $\omega \in \mathbb{R}^m$ is the angular frequency in the Fourier domain, and $F_n$ is the $n$-dimensional Fourier transformation. The Cuckoo Search algorithm consists of two steps: the local search and the global search. At the local search, one member of population (representing cuckoo nest), say $x^{(i)} \in \mathcal{P}$, is chosen and from this position the Levy flight is performed to create a new member $y$

$$y = x^{(i)} + \theta \xi,$$

where $\theta$ is a scaling parameter, which should be related to the size of the search space, and $\xi \sim L(\alpha, n)$. In this work, we are using $\theta = \frac{\|\Omega\|}{10^4}$, where the norm of the search space $\Omega = \bigotimes_i [a_i, b_i]$ is defined as

$$\|\Omega\| = \max_i |b_i - a_i|$$

If $TPD(y;x^*) < TPD(x^{(k)};x^*)$, where $k \sim U_d(1, NP)$, then $x^{(k)}$ is replaced in the population $\mathcal{P}$ by $y$. At the global search, the population is sorted with respect to its TPD values and worst $p$ percent of the population are replaced with new ones using Levy flights, i.e.,

$$x^{(k)} = x^{(k)} + \theta \xi,$$

where $\xi \sim L(\alpha, n)$ and $k > (1-p)NP$. These two steps are repeated until a stopping criterion is met. Suggested values from the literature are $\alpha = 1$, population size $NP = 15$ and $p = 0.25$. These values are used in our calculations. The summary of this algorithm is presented in Algorithm 2.

### 3.4. Harmony Search.

The third heuristic for the comparison is the Harmony Search, which was created by Geem [5]. Among the presented algorithms, the Harmony Search is the simplest one. In each iteration, a new member $y$ of the population is created using “improvisation”. In this technique, each component $y_k$ of the new member $y = (y_1, \ldots, y_n)^T$ is created using the current population or is chosen randomly. First, random number $i \sim U_c(0, 1)$ is generated. If $i$ is greater or equal than a given parameter $HCMR$ then $y_k$ is chosen randomly in the search space. On the other hand, if $i < HCMR$ then the $k$-th component is copied from randomly chosen member of the population, i.e., $y_k = x^{(j)}_k$, where $j \sim U_d(1, NP)$. Moreover, in the second situation, a random number $o \sim U_c(0, 1)$ is generated and if $o$ is lower than a given parameter $PAR$, than the solution is perturbed using equation

$$y_k = y_k + BW \cdot \Delta,$$

where $BW$ is given parameter and $\Delta \sim U_c(-1,1)$. In our computations, we are using suggested values [5] $HCMR = 0.85$ (harmony memory consideration rate), $PAR = 0.5$ (pitch adjusting rate) and $BW = 0.001$ (bandwidth). The pseudo code
of the “improvisation” technique is provided in Algorithm 3. After the improvisation, if the new member \( y \) has lower value of the objective function than the worst member of the population \( P \), then the worst member of the population is replaced in the population. These steps are repeated until a stopping criterion is met. The summary of the Harmony Search algorithm is presented in Algorithm 4.

In our calculations, the size of the population is set to suggested value \( NP = 6 \) [5]. Therefore, compared to the size of the population in the Differential Evolution or the Cuckoo Search, the population in the Harmony Search is considerably smaller.

### 3.5. Covariant Matrix Adaptation

In the Covariant Matrix Adaptation (CMA-ES) [15, 6], the next generation of the population is generated from the normal distribution

\[
x_i^{(g+1)} \sim m^{(g)} + \sigma^{(g)} \mathcal{N}(0, C^{(g)}),
\]

where \( m^{(g)} \) is the mean value, \( C^{(g)} \) is the covariance matrix, and \( \sigma^{(g)} \) is the step size of the current generation. The mean value of the next generation \( m^{(g+1)} \) is calculated using

\[
m^{(g+1)} = m^{(g)} + c_m \sum_{i=1}^{\mu} w_i \left( x_i^{(g+1)} - m^{(g)} \right)
\]

where \( \mu \) is the number of parents, \( c_m \) is the learning rate and \( w_i \) are the weights, which have to satisfy \( \sum_{i=1}^{\mu} w_i = 1 \). The covariance matrix \( C \) is updated using

\[
C^{(g+1)} = (1 - 2c_\sigma)C^{(g)} + c_\mu \sum_{i=1}^{\mu} w_i y_i y_i^T + c_p \left( p^{(g+1)}_c \right)^T,
\]

where \( p_c \) and \( y_i \) are calculated using

\[
p^{(g+1)}_c = (1 - c_\sigma) p^{(g)}_c + \sqrt{c_c (2 - c_c) \mu_{eff}} \frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}},
\]

\[
y_i = \frac{1}{\sigma^{(g)}} \left( x_i^{(g+1)} - m^{(g)} \right).
\]

Lastly, the step size \( \sigma^{(g)} \) is updated using

\[
\sigma^{(g+1)} = \sigma^{(g)} \exp \left\{ \epsilon_\sigma \left( \frac{||p^{(g+1)}_c||}{E||\mathcal{N}(0, I)||} - 1 \right) \right\},
\]

where

\[
p^{(g+1)}_c = (1 - c_\sigma) p^{(g)}_c + \sqrt{c_c (2 - c_c) \mu_{eff}} C^{-\frac{1}{2}} \frac{m^{(g+1)} - m^{(g)}}{\sigma^{(g)}},
\]

and \( E||\mathcal{N}(0, I)|| \) is estimated using [7]

\[
E||\mathcal{N}(0, I)|| = \sqrt{n} \left( 1 - \frac{1}{4n} + \frac{1}{21n^2} \right).
\]

In the previous equations, \( c_1, c_\mu, c_\sigma, d_\sigma, c_c, \) and \( \mu_{eff} \) are numerical constants. In this work, we have taken the numerical values from [7]. The size of the population is set to \( NP = 8 \) and \( \mu = 4 \). The pseudo code of the CMA-ES algorithm is provided in Algorithm 5.
3.6. Elephant Herding Optimization. In the EHO algorithm, the population of size $NP$ is divided into $NC$ clans. Let denote $NE_k$ the number of elephants in the $k$-th clan, $x^{(k,i)}$ the $i$-th elephant in the $k$-th clan, and let $x^{(k,1)}$ is the elephant with the lowest value of the objective function in the $k$-th clan. Then, the next generation of the population is created using

$$
 x^{(k,i)} = x^{(k,1)} + \delta \left( x^{(k,1)} - x^{(k,i)} \right) r,
$$

(22)

for $i \in \hat{NE}_k \setminus \{1\}$ and $k \in \hat{N}C$. In the previous equation, $\delta \in (0, 1)$ is a scale factor and $r \sim U_c(0, 1)$. The best elephants in each clan $x^{(k,1)}$ are updated using

$$
 x^{(k,1)} = \frac{\beta}{NE_k} \sum_{j=1}^{NE_k} x^{(k,j)},
$$

(23)

where $\beta$ is a scale factor. The second step of the algorithm is the separation of the worst member of the population. In each clan, the member with the highest value of the objective function is replaced in the population with a randomly created solution. Lastly, the algorithm includes an elitism strategy. At the beginning of each iteration, the best $\gamma$ members of the population are stored, and at the end of the iteration, replace the worst $\gamma$ members of the population. The pseudo code of the EHO algorithm is provided in Algorithm 6. In this work, we are using values $\delta = 0.5$, $\beta = 0.1$, $NP = 30$, $NC = 5$, and $\gamma = 2$.

3.7. Mirroring. All presented heuristics are designed to find the global minimum of a function in a rectangular domain defined by $\Omega = \bigotimes_{i=1}^n [a_i, b_i]$, where $a = (a_1, \ldots, a_n)^T$ and $b = (b_1, \ldots, b_n)^T$ define the boundary of the domain $\Omega$. In the case of the VTN-specification of the phase stability testing $a = (0, \ldots, 0)^T$ and $b = \left( \frac{1}{\bar{b}_1}, \ldots, \frac{1}{\bar{b}_n} \right)^T$, where $b_k$ for $k \in \hat{n}$ is the co-volume parameter of the Peng-Robinson equation of state (see the Appendix for more details). In many cases it may happen that the new member of the population (mutant, cuckoo, improvisation) is created outside of this domain $\Omega$ or outside the feasible domain $D \subset \Omega$. To overcome this situation, a method called mirroring is adopted to our algorithms. If the new member of the population will be placed outside of $\Omega$, it will be bounced back from the boundary to the domain $\Omega$. This method is well-known and used in many optimization algorithms. However, since in the phase stability problems, the feasible domain is a simplex $D \subset \Omega$, using only the mirroring technique will not be satisfactory. Principally, three possible techniques can be used:

- penalty function in the unfeasible area,
- redefinition of the objective function outside the simplex,
- generalization of the mirroring technique to handle simplex geometry.

The first possibility can not be used alone since the objective function is not defined outside the feasible area. In the second technique, we can define TPD function as a big positive constant outside the feasible simplex $D$. However, in our experiments, this technique did not provide satisfactory results. Therefore, we expanded the mirroring technique to mirror a vector into a simplex. The solution $y = (y_1, \ldots, y_n)^T$ of the phase stability testing has to be in a simplex $D$ defined by equations

$$
 y_k \geq 0, \quad k \in \hat{n}
$$

(24)
\[
\sum_{i=1}^{n} b_i y_i \leq 1 - \varepsilon, \tag{25}
\]

where \( \varepsilon > 0 \) is a small positive number as the solution has to satisfy \( \sum_{i=1}^{n} b_i c_i < 1 \) (see the Appendix). In this work, we are using \( \varepsilon = 10^{-6} \). In the first step of our algorithm, we transform the solution with relation \( x_k = \frac{(1-\varepsilon)y_k}{b_k} \) for \( k \in \hat{n} \). Now, the problem is to mirror the transformed solution \( x = (x_1, \ldots, x_n)^T \) into the unit simplex defined by

\[
x_k \geq 0, \quad k \in \hat{n} \tag{26}
\]

\[
\sum_{i=1}^{n} x_i \leq 1. \tag{27}
\]

In the second step, the solution is mirrored to satisfy \( x \in \otimes_{i \in \hat{n}} [0,1] \). This can be done using the standard mirroring technique. If (27) does not hold, the next task is to find a vector \( z = (z_1, \ldots, z_n)^T \) which is on the boundary of the simplex (26)-(27) and its distance from the solution \( x \) is minimal. Therefore, the vector \( z \) has to satisfy

\[
\sum_{i=1}^{n} z_i = 1, \tag{28}
\]

\[
\frac{1}{2} \| x - z \|_2^2 \rightarrow \min, \tag{29}
\]

\[
z_k \geq 0, \quad k \in \hat{n}, \tag{30}
\]

where the norm in (29) is the Euclidean norm. The factor \( \frac{1}{2} \) in (29) simplifies derivation and does not change the resulting vector \( z \). The vector \( z \) can be found using Lagrange multipliers method and reads as

\[
z_k = x_k - \frac{1}{n} \left( \sum_{i=1}^{n} x_i - 1 \right), \quad k \in \hat{n}. \tag{31}
\]

The resulting novel mirroring process begins with \( x \in \Omega \) and terminates for \( x^{(\text{new})} \), which satisfies equations (26)-(27), using iteration scheme

\[
x^{(\text{new})} = |x - 2z| = \left| x - \frac{2}{n} \left( \sum_{i=1}^{n} x_i - 1 \right) \right|, \tag{32}
\]

which terminates in a finite number of steps. In equation (32), \( 1 = (1, \ldots, 1)^T \in \mathbb{R}^n \), and the absolute value is applied component-wise. In the last step of our algorithm, after a solution \( x^{(\text{new})} \) has been found, the reverse transformation \( y^{(\text{new})}_k = \frac{x^{(\text{new})}_k}{1 - \varepsilon} b_k \) is applied to create \( y^{(\text{new})} \in D \).

4. Results. We have implemented the presented algorithms in our C++ GeneralFlashSolver library [26] and compare the algorithms on problems from the literature. For the computation, a computer with processor Intel i7-4770 3.40 GHz has been used. First, the Newton-Raphson method was used to calculate the solution \( x^{(\text{opt})} \). The Newton-Raphson iterations were stopped when

\[
\| \Delta y^{(k)} \| < 10^{-6}, \tag{33}
\]
Algorithm 1: Differential Evolution

**Input**: \( NP > 0, F_{\min} \in (0, 1], F_{\max} \in (0, 1], CR_{\min} \in (0, 1], CR_{\max} \in (0;1] \)

1. randomly initialize population \( P = \{ x^{(i)}; i \in \mathcal{N}P \} \),
2. while stop criterion do
   3. update values of \( F \) and \( CR \) using equations (6) and (8)
   4. for all members \( x^{(i)} \in P \) do
      5. create a mutant member \( y \) using equation (5)
      6. create a crossover member \( z \) from \( x^{(i)} \) and \( y \) using equation (7)
      7. if \( TPD(z; x^*) < TPD(x^{(i)}; x^*) \) then
         8. replace \( x^{(i)} \) in the population with \( z \)
   9. end
10. end
11. end

Algorithm 2: Cuckoo Search

**Input**: \( NP > 0, p \in (0;1), \alpha \in (0;2), \theta \in \mathbb{R}^+ \)

1. randomly initialize population \( P = \{ x^{(i)}; i \in \mathcal{N}P \} \),
2. while stop criterion do
   3. choose randomly \( j \in U_d(1, NP) \)
   4. create a new member \( y \) using equation (10)
   5. choose randomly \( k \in U_d(1, NP) \)
   6. if \( TPD(y; x^*) < TPD(x^{(k)}; x^*) \) then
      7. replace \( x^{(k)} \) in the population with \( y \)
   8. end
   9. sort the population with respect to the TPD value
10. replace worst \( p \) percent of the population using Levy flight
    \[ x^{(k)} = x^{(k)} + \theta \xi, \quad \xi \sim L(\alpha, n) \]
    for \( k > (1 - p)NP \)
11. end

**Figure 1.** Geometric interpretation of mirroring into the feasible simplex.
Algorithm 3: Harmony Search improvisation

**Input**: $HMCR \in (0, 1), \ PAR \in (0, 1), \ BW \in (0, 1)$

**Output**: $y = (y_1, \ldots, y_n)^T$

1. for $k = 1, \ k \leq n, \ k \ ++ \ do$
2. randomly choose $i \in U_c(0, 1)$
3. if $i < HMCR$ then
4. randomly choose $j \in U_d(1, NP)$
5. set $k$-th component of the $y$ using: $y_k = x^{(j)}_k$
6. randomly choose $o \in U_c(0, 1)$
7. if $o < PAR$ then
8. randomly choose $\Delta \in U_c(-1, 1)$
9. reset $k$-th component using: $y_k = y_k + BW \cdot \Delta$
10. else
11. set $k$-th component of the $y$ randomly inside feasible area
12. $y_k = U_c(a_k, b_k)$
13. end
14. end

Algorithm 4: Harmony Search

**Input**: $NP > 0, \ HMCR \in (0, 1), \ PAR \in (0, 1), \ BW \in (0, 1)$

1. randomly initialize population $\mathcal{P} = \{x^{(i)}; i \in \hat{NP}\}$,
2. while stop criterion do
3. randomly choose $j \in U_d(1, NP)$
4. create a new member $y$ using improvisation from Algorithm 3
5. if $\text{TPD}(y;x^*) < \text{TPD}(x^{(\text{worst})};x^*)$ then
6. replace $x^{(\text{worst})}$ in the population with $y$
7. end
8. end

Algorithm 5: CMA-ES

**Input**: $NP > 0, \mu > 0, \ c_1, \ c_\mu, \ c_\sigma, \ c_c$, and $\mu_{eff}$

1. randomly initialize population $\mathcal{P} = \{x^{(i)}; i \in \hat{NP}\}$,
2. initialize the mean and the covariance matrix
3. while stop criterion do
4. for all members $x^{(i)} \in \mathcal{P}$ do
5. update a population member $x^{(i)}$ using equation (14)
6. end
7. update the mean using equation (15)
8. update the covariance matrix using equation (16)
9. update the step size using equation (19)
10. end
Algorithm 6: Elephant Herding Optimization

Input : \( NP > 0, NC > 0, \gamma > 0, \delta, \beta \in (0, 1] \)

1. randomly initialize population \( P = \{ x^{(k,i)}; k \in \mathcal{NC}, i \in \mathcal{NE}_k \} \),

2. while stop criterion do
   3. store the best \( \gamma \) member of the population
   4. divide the population into clans
   5. for all members \( x^{(k,i)} \in P \) do
      6. if \( i == 1 \) then
         7. update \( x^{(k,1)} \) using equation (23)
      8. end
      9. else
         10. update member \( x^{(k,i)} \) using equation (22)
      11. end
   12. replace the worst member in each clan with randomly created
   13. replace the \( \gamma \) worst members with the stored ones
   14. end
15. end

| Component | \( T_{\text{crit}} [\text{K}] \) | \( P_{\text{crit}} [\text{MPa}] \) | \( \omega [-] \) |
|-----------|----------------------|----------------------|-------------|
| \( C_1 \) | 190.40               | 4.60                 | 0.0110      |
| \( C_3 \) | 369.80               | 4.25                 | 0.1530      |
| \( \text{CO}_2 \) | 304.14               | 7.375                | 0.2390      |
| \( \text{N}_2 \) | 126.21               | 3.390                | 0.0390      |
| \( \text{PC}_1 \) | 333.91               | 5.329                | 0.1113      |
| \( \text{PC}_2 \) | 456.25               | 3.445                | 0.2344      |
| \( \text{PC}_3 \) | 590.76               | 2.376                | 0.4470      |
| \( C_{12+} \) | 742.58               | 1.341                | 0.9125      |

Table 2. Parameters of the Peng-Robinson equation of state used in Examples 1–2.

where the norm \( \|\cdot\| \) is defined in [26] as

\[
\|y\|^2 = \|(c_1, \ldots, c_n)^T\|^2 = \sum_{i=1}^{n} \frac{c_i^2}{(c_i^*)^2}.
\]  

(34)

In the previous equation, \( c_i^* \) is the initial concentration of the \( i \)-th component. In all cases, the found minimum \( x^{(opt)} \) was the correct global minimum. This was proven by the global optimization method based on the Branch and Bound strategy [27]. Having the correct solution at hand, the evolution algorithms are stopped with success if the found solution \( y \) satisfies

\[
\|y - x^{(opt)}\| < 10^{-5} \text{ or } \frac{\text{TPD}(y;x^*) - \text{TPD}(x^{(opt)};x^*)}{\text{TPD}(x^{(opt)};x^*) + 1} < 10^{-5}.
\]  

(35)

The maximum number of function evaluations for each evolution algorithm was set to \( FE_{\text{max}} = 5 \times 10^4 \), and each algorithm was run 100 times. Physical parameters of the equation of state of all used components are presented in Table 2.
Example 1. In the first example, we investigate a binary mixture of methane ($C_1$) and propane ($C_3$) with mole fractions $z_{C_1} = 0.547413$, $z_{C_3} = 0.452587$, temperature in range $T^* \in [250;350]$ K, and total concentration in range $c^* \in (0;25000]$ mol m$^{-3}$. The binary interaction coefficient is $\delta_{C_1-C_3} = 0.0365$. In Figure 2a, the minima of the TPD function in the $cT$-space are presented. A grid with $51 \times 51$ points was used, therefore, total 2601 phase stability testing calculations (minimization) were performed. In Figure 2a, the red line represents the phase boundary. Above this line, the state is stable, and the global minimum has value TPD = 0. In Figures 2b–2f, the numbers of successful runs for each evolution algorithm are presented. The most successful algorithm was the Differential Evolution, which found the correct solution in almost all cases. The CMA-ES correctly found solution in most of the cases. However, the algorithm had problems at the phase boundary and in an area with total concentration $c^* = 10^4$ mol m$^{-3}$. In this area, the other heuristics also had problems to find the correct solution. The Cuckoo Search was able to find the correct solution almost everywhere in the unstable area. However, in the stable area, the algorithm was able to find the correct solution in one part of the stable area and only with the probability around 70 %. The performance of the last two algorithms (HS and EHO) was not that satisfactory. Neither of them finds the correct solution in the stable area, and in the unstable area, they find the correct solution at best at 20 % of the runs. In our opinion, this was caused by the value of the maximum number of iteration as the HS and EHO algorithms have slower convergence.

Example 2. In the second example, we investigate a seven component mixture of nitrogen ($N_2$), carbon dioxide ($CO_2$), methane ($C_1$) and four hydrocarbon pseudo-components denoted as $PC_1$, $PC_2$, $PC_3$ and $C_{12+}$ with mole fractions $z_{N_2} = 0.466905$, $z_{CO_2} = 0.007466$, $z_{C_1} = 0.300435$, $z_{PC_1} = 0.105051$, $z_{PC_2} = 0.041061$, $z_{PC_3} = 0.045060$, and $z_{C_{12+}} = 0.034021$. The binary interaction coefficients between all components are presented in Table 3. We investigate the phase stability in temperature range $T^* \in [250;650]$ K and the whole range of feasible total concentrations $c^*$. In Figure 3a, the minima of the TPD function in the $cT$-space are presented. As in Example 1, a grid with $51 \times 51$ points was used, which resulted in 2601 phase stability calculations. In Figures 3b–3f, the numbers of successful runs for each evolution algorithm are presented. Similarly to the first example, the Harmony Search, Cuckoo Search, and Elephant Herding Optimization did not perform satisfactorily. The Differential Evolution and CMA-ES find the correct solution in most cases. However, both algorithms had problems in some areas. The Differential Evolution encountered problems when the total concentration $c^*$ was higher than $1.7 \times 10^4$ mol m$^{-3}$, the CMA-ES algorithm if the temperature was lower than 300 K and the total concentration was approximately $1.5 \times 10^4$ mol m$^{-3}$.

Comparison. To compare the evolution strategies, the Wilcoxon singed-rank test [9] was performed with the criterion [1, 10]

$$CRIT = \left(\frac{1-p_s}{p_s}\right) FE_{max} + \mathbb{E}(n_e),$$

(36)

where $p_s$ is the probability of the success and $\mathbb{E}(n_e)$ is the mean value of the function evaluation of the successful runs. This criterion is calculated for both previous examples, therefore, we have vectors of size 5202 for each evolution algorithm. Then, a pairwise Wilcoxon test was performed using Matlab software. All hypotheses
Component | N₂ | CO₂ | C₁ | PC₁ | PC₂ | PC₃ | C₁₂+
--- | --- | --- | --- | --- | --- | --- | ---
N₂ | 0.000 | 0.000 | 0.100 | 0.100 | 0.100 | 0.100 | 0.100
CO₂ | 0.000 | 0.000 | 0.150 | 0.150 | 0.150 | 0.150 | 0.150
C₁ | 0.100 | 0.150 | 0.000 | 0.035 | 0.040 | 0.049 | 0.069
PC₁ | 0.100 | 0.150 | 0.035 | 0.000 | 0.000 | 0.000 | 0.000
PC₂ | 0.100 | 0.150 | 0.040 | 0.000 | 0.000 | 0.000 | 0.000
PC₃ | 0.100 | 0.150 | 0.049 | 0.000 | 0.000 | 0.000 | 0.000
C₁₂+ | 0.100 | 0.150 | 0.069 | 0.000 | 0.000 | 0.000 | 0.000

Table 3. The binary interaction coefficients between all components in Example 2.

| | Example 1 | Example 2 |
|---|---|---|
| Newton-Raphson | 0.99 | 11.55 |
| Differential Evolution | 35.87 | 995.03 |
| Cuckoo Search | 78.55 | 394.63 |
| Harmony Search | 210.72 | 862.48 |
| CMA-ES | 26.48 | 408.91 |
| Elephant Herding Optimization | 500.30 | 1777.72 |

Table 4. Computation times in seconds for Examples 1–2.

about the median equality have been rejected on the significance level $\alpha = 0.05$. In all ten tests, the calculated $p$-values have been lower than $10^{-9}$. In addition, we obtained same results with the Bonferroni correction [4]. Moreover, in Table 4, the computation times for all algorithms, including the Newton-Raphson method, are presented. The computation times of the evolution algorithms are the mean value for the 100 runs. From this point of view, the Newton-Raphson method is the best method as the computation time is about twenty times faster than the fastest evolution heuristic. However, as the computations are stopped when the correct solution has been found, the HS and EHO algorithms have to do more iterations (evaluations) than more successful algorithms (DE or CMA-ES). Therefore, the comparison of the computation times of HS or EHO with DE or CMA-ES is not legitimate. The comparison of DE or CMA-ES with the Newton-Raphson method is valid as all algorithms found the correct solution in most cases.

5. **Summary and conclusions.** In this work, we compared different optimization methods for the phase stability testing of multicomponent mixtures. We compared the performance of five different evolution strategies: Differential Evolution, Cuckoo Search, Harmony Search, CMA-ES, and Elephant Herding Optimization. Based on the given examples, the best performance had the Differential Evolution and CMA-ES algorithm. The other three algorithms had not been able to find the solution in most cases. Moreover, we performed the Wilcoxon signed-rank test to prove the significant difference between the methods. All pairwise tests resulted with the rejection of the hypothesis about the median equality. The highest $p$-value was less than $10^{-9}$. However, none of the evolution strategies found the solution in 100 percent of cases. Therefore, the usage of these algorithms in the area of chemical engineering is limited. Even more, in comparison with the classical Newton-Raphson method with line-search, the computation times of the evolution strategies were significantly higher.
Figure 2. Global minimum of the TPD function in the $cT$-space and the number of successful runs of each evolution heuristic. The red line represents the phase boundary (above the line the global minimum is zero). Example 1: mixture C$_1$-C$_3$. 
Figure 3. Global minimum of the TPD function in the $cT$-space and the number of successful runs of each evolution heuristic. The red line represents the phase boundary (above the line the global minimum is zero). Example 2: mixture $N_2$-$CO_2$-$C_1-PC_i$. 
Appendix. We are using Peng-Robinson equation of state [18] in the form

\[
P^{(PR)}(c_1, \ldots, c_n) = cRT \left( 1 - \frac{bc}{1 + 2bc - b^2c^2} \right) + \frac{ac^2}{1 - 2bc},
\]

where \( T \) is temperature, \( R \) is universal gas constant, \( c = \sum_{i=1}^{n} c_i \) is total concentration, and parameters \( a, b \) can be calculated using

\[
a = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j a_{ij},
\]

\[
a_{ij} = (1 - \delta_{i-j}) \sqrt{a_i a_j},
\]

\[
a_i = 0.45724 \frac{RT_{i,\text{crit}}^2}{P_{i,\text{crit}}^2} \left[ 1 + m_i \left( 1 - \sqrt{T_{i,r}} \right) \right]^2,
\]

\[
b = \sum_{i=1}^{n} x_i b_i,
\]

\[
b_i = 0.0778 \frac{RT_{i,\text{crit}}}{P_{i,\text{crit}}},
\]

\[
m_i = \begin{cases} 
0.37464 + 1.54226 \omega_i - 0.26992 \omega_i^2 & \text{for } \omega_i < 0.5, \\
0.3796 + 1.485 \omega_i - 0.1644 \omega_i^2 + 0.01667 \omega_i^3 & \text{for } \omega_i \geq 0.5.
\end{cases}
\]

In the equations above, \( x_i = c_i/c \) denotes the molar fraction of component \( i \), \( \delta_{i-j} \) is the binary interaction parameter between components \( i \) and \( j \), \( T_{i,\text{crit}}, P_{i,\text{crit}} \) are the critical temperature and the critical pressure of component \( i \), respectively, \( T_{i,r} \) is the reduced temperature defined as \( T_{i,r} = \frac{T}{T_{i,\text{crit}}} \) and \( \omega_i \) is the acentric factor of component \( i \). For the Peng-Robinson equation of state, the feasible domain \( D \) reads as

\[
D = \left\{ (c_1, \ldots, c_n)^T : \sum_{k=1}^{n} b_k c_k < 1 \land c_i \geq 0, i \in \hat{n} \right\}.
\]

The Helmholtz density function \( a \) is calculated using equation

\[
a(c_1, \ldots, c_n) = \frac{1}{c} \int_{c_0}^{c} \frac{1}{(c')^c} P \left( \frac{c_1}{c}, \ldots, \frac{c_n}{c}, c' \right) dc',
\]

where \( c_0 > 0 \) is total concentration which satisfies

\[
a(c_0 z_1, \ldots, c_0 z_n) = 0,
\]

where \( z_i \) is mole fraction of \( i \)-th component. With the Peng-Robinson equation of state, the integral in equation (44) can be analytically calculated. Therefore, the final equation for the Helmholtz density function reads as

\[
a^{(PR)}(c_1, \ldots, c_n) = RT \left( \sum_{i=1}^{n} c_i \ln \frac{c_i}{c_0} - c \ln(1 - bc) \right) + \frac{ac}{2\sqrt{2b}} \ln \left[ \frac{1 + (1 - \sqrt{2}b)c}{1 + (1 + \sqrt{2}b)c} \right].
\]

Lastly, using

\[
\mu_i(c_1, \ldots, c_n) = \frac{\partial a}{\partial c_i}(c_1, \ldots, c_n),
\]
the chemical potential of \(i\)-th component \(\mu_i\) for \(i \in \hat{n}\) reads as
\[
\mu_i^{(PR)}(c_1, \ldots, c_n) = RT \left( \ln \frac{c_i}{c_0} + 1 - \ln(1 - bc) + \frac{cb_i}{1 - bc} \right) - \frac{acb_i}{b} \frac{1}{1 + 2bc - (bc)^2} + \frac{1}{\sqrt{2bc}} \ln \frac{|1 + (1 - \sqrt{2})bc|}{|1 + (1 + \sqrt{2})bc|}. \tag{48}
\]

Acknowledgments. The work was supported by the project no. 17–06759S of the Czech Science Foundation and the project no. SGS17/196/OHK4/3T/14 of the Czech Technical University in Prague. The author JK acknowledges the support of the OP VVV MEYS funded project CZ.02.1.01/0.0/0.0/16_019/0000765 “Research Center for Informatics”.

REFERENCES

[1] A. Auger and N. Hansen, Performance evaluation of an advanced local search evolutionary algorithm, 2005 IEEE Congress on Evolutionary Computation, 2 (2005), 1777–1784.
[2] R. Bellman, Dynamic Programming, Princeton University Press, Princeton, N. J., 1957.
[3] J. Brest, S. Greiner, B. Bosković, M. Mernik and V. Žumer, Self-adapting control parameters in Differential Evolution: A comparative study on numerical benchmark problems, IEEE Transactions on Evolutionary Computation, 10 (2006), 646–657.
[4] O. Dunn, Multiple comparisons among means, Journal of the American Statistical Association, 56 (1961), 52–64.
[5] Z. W. Geem, Music-Inspired Harmony Search Algorithm: Theory and Applications, Springer, 2001.
[6] N. Hansen and A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evolutionary Computation, 9 (2001), 159–195.
[7] N. Hansen, The CMA evolution strategy: A tutorial, CoRR, URL http://arxiv.org/abs/1604.00772.
[8] T. Jindrová and J. Mikyška, General algorithm for multiphase equilibria calculation at given volume, temperature, and moles, Fluid Phase Equilibria, 393 (2015), 7–25.
[9] W. H. Kruskal, Historical notes on the Wilcoxon unpaired two-sample test, Journal of the American Statistical Association, 52 (1957), 356–360.
[10] J. Kukal and M. Mojžiš, Quantile and mean value measures of search process complexity, Journal of Combinatorial Optimization, 35 (2018), 1261–1285.
[11] M. L. Michelsen, The isothermal flash problem. Part I. Stability, Fluid Phase Equilibria, 9 (1982), 1–19.
[12] M. L. Michelsen, The isothermal flash problem. Part 2. Phase-split computation, Fluid Phase Equilibria, 9 (1982), 21–40.
[13] J. Mikyška and A. Firoozabadi, Investigation of mixture stability at given volume, temperature, and moles, Fluid Phase Equilibria, 321 (2012), 1–9.
[14] A. W. Mohamed, H. Z. Sabry and M. Khorsheid, An alternative differential evolution algorithm for global optimization, Journal of Advanced Research, 3 (2012), 149–165.
[15] A. Ostermeier, A. Gawelczyk and N. Hansen, A derandomized approach to self-adaptation of evolution strategies, Evolutionary Computation, 2 (1994), 369–380.
[16] I. Pavlyukevich, Cooling down Lévy flights, Journal of Physics: A Mathematical and Theoretical, 40 (2007), 12299–12313.
[17] I. Pavlyukevich, Lévy flights, non-local search and simulated annealing, Journal of Computational Physics, 226 (2007), 1830–1844, URL http://www.sciencedirect.com/science/article/pii/S002199910700263X.
[18] D.-Y. Peng and D. B. Robinson, A new two-constant equation of state, Industrial & Engineering Chemistry Fundamentals, 15 (1976), 59–64.
### List of symbols

**Functions:**
- $a$ Helmholtz density function
- $g$ probability density function of $\alpha$-stable distribution
- $L(\alpha, n)$ $\alpha$-stable n-dimension distribution
- $P$ pressure function
- $U_c(a, b)$ uniform continuous distribution from $a$ to $b$
- $U_d(a, b)$ uniform discrete distribution from $a$ to $b$
- $\mu_i$ chemical potential of $i$-th component

**Parameters:**
- $a, b$ parameters of the Peng-Robinson equation of state
- $BW$ bandwidth in the Harmony Search
- $CR$ crossover rate in the Differential Evolution
- $F$ scaling parameter in the Differential Evolution
- $HMCR$ harmony memory consideration rate
- $PAR$ pitch adjusting rate in the Harmony Search
- $NP$ the population size
- $\alpha$ parameter of $\alpha$-stable distribution
- $\theta$ scaling factor in Cuckoo Search
- $\beta, \delta$ parameter in the EHO

**Search space:**
- $D$ feasible area for phase stability testing
- $n$ dimension of the optimization problem
- $P$ population
- $\Omega$ rectangular domain

**Other:**
- $c$ total concentration
- $c_i$ concentration of $i$-th component
- $g$ gradient of the TPD function
- $H$ Hessian matrix of the TPD function
- $P_{\text{crit}}$ critical pressure
- $T$ temperature
- $T_{\text{crit}}$ critical temperature
- $x^*$ vector describing initial phase
- $\lambda$ damping factor in the Newton-Raphson method
- $\omega$ acentric factor
- $\tilde{k}$ \{1, \ldots, $k$\}

**Superscripts:**
- $T$ transposition
- $*$ initial phase indicator

---

[19] O. Polívka and J. Mikyška, Compositional modeling in porous media using constant volume flash and flux computation without the need for phase identification, *Journal of Computational Physics*, 272 (2014), 149–179.

[20] K. V. Price, R. M. Storn and J. A. Lampinen, *Differential Evolution. A Practical Approach to Global Optimization*, With 1 CD-ROM (Windows, Macintosh and UNIX). Natural Computing Series. Springer-Verlag, Berlin, 2005.

[21] A. K. Qin, V. L. Huang and P. N. Suganthan, Differential Evolution algorithm with strategy adaptation for global numerical optimization, *IEEE Transactions on Evolutionary Computation*, 13 (2009), 398–417.
[22] A. M. Reynolds and M. A. Frye, Free-flight odor tracking in drosophila is consistent with an optimal intermittent scale-free search, PloS ONE, 2 (2007), e354.
[23] R. M. Rizk-Allah, R. A. El-Sehiemy and G.-G. Wang, A novel parallel hurricane optimization algorithm for secure emission/economic load dispatch solution, Applied Soft Computing, 63 (2018), 206–222, URL http://www.sciencedirect.com/science/article/pii/S1568494617307160.
[24] R. B. Schnabel and E. Eskow, A revised modified Cholesky factorization algorithm, SIAM Journal on Optimization, 9 (1999), 1135–1148.
[25] T. Smejkal and J. Mikyška, Phase stability testing and phase equilibrium calculation at specified internal energy, volume, and moles, Fluid Phase Equilibria, 431 (2017), 82–96.
[26] T. Smejkal and J. Mikyška, Unified presentation and comparison of various formulations of the phase stability and phase equilibrium calculation problems, Fluid Phase Equilibria, 476 (2018), 61–88.
[27] T. Smejkal and J. Mikyška, VTN-phase stability testing using the branch and bound strategy and the convex-concave splitting of the Helmholtz free energy density, Fluid Phase Equilibria, 504 (2020), 112323, URL http://www.sciencedirect.com/science/article/pii/S037838121930384X.
[28] R. Storn and K. Price, Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization, 11 (1997), 341–359.
[29] G. Wang and Y. Tan, Improving metaheuristic algorithms with information feedback models, IEEE Transactions on Cybernetics, 49 (2019), 542–555.
[30] G.-G. Wang, S. Deb and L. dos S. Coelho, Elephant herding optimization, 3rd International Symposium on Computational and Business Intelligence, 2015.
[31] G.-G. Wang, A. Gandomi, X.-S. Yang and A. Alavi, A new hybrid method based on Krill Herd and Cuckoo Search for global optimisation tasks, International Journal of Bio-Inspired Computation, 8 (2016), 286–299.
[32] G.-G. Wang, A. H. Gandomi and A. H. Alavi, An effective krill herd algorithm with migration operator in biogeography-based optimization, Applied Mathematical Modelling, 38 (2014), 2454–2462, URL http://www.sciencedirect.com/science/article/pii/S0307904X13006756.
[33] G.-G. Wang, A. H. Gandomi, A. H. Alavi and G.-S. Hao, Hybrid krill herd algorithm with differential evolution for global numerical optimization, Neural Computing and Applications, 25 (2014), 297–308.
[34] G.-G. Wang, L. H. Guo, A. H. Gandomi, G.-S. Hao and H. Q. Wang, Chaotic krill herd algorithm, Information Sciences, 274 (2014), 17-24.
[35] G. Wang, L. Guo, H. Wang, H. Duan, L. Liu and J. Li, Incorporating mutation scheme into Krill Herd algorithm for global numerical optimization, Neural Computing and Applications, 24 (2014), 853–871, URL https://doi.org/10.1007/s00521-012-1304-8.
[36] H. Q. Wang and J.-H. Yi, An improved optimization method based on krill herd and artificial bee colony with information exchange, Memetic Computing, 10 (2018), 177–198.
[37] Y. Q. Wu and Z. M. Chen, The application of high-dimensional sparse grids in flash calculations: From theory to realisation, Fluid Phase Equilibria, 464 (2018), 22–31.
[38] Y. Q. Wu, C. Kowitz, S. Y. Sun and A. Salama, Speeding up the flash calculations in two-phase compositional flow simulations - The application of sparse grids, Journal of Computational Physics, 285 (2015), 88–99.
[39] Y. Q. Wu and M. Q. Ye, A parallel sparse grid construction algorithm based on the shared memory architecture and its application to flash calculations, Computers & Mathematics with Applications, 77 (2019), 2114–2129, URL http://www.sciencedirect.com/science/article/pii/S0898122118307004.
[40] X.-S. Yang, Cuckoo search via Levy flights, World Congress on Nature Biologically Inspired Computing, (2009), 210–214.
[41] J.-H. Yi, S. Deb, J. Y. Dong, A. H. Alavi and G.-G. Wang, An improved NSGA-III algorithm with adaptive mutation operator for big data optimization problems, Future Generation Computer Systems, 88 (2018), 571–585, URL http://www.sciencedirect.com/science/article/pii/S0167739X18306137.
[42] J.-H. Yi, L.-N. Xing, G.-G. Wang, J. Dong, A. V. Vasilakos, A. H. Alavi and L. Wang, Behavior of crossover operators in NSGA-III for large-scale optimization problems, Information Sciences, 509 (2020), 470–487, URL http://www.sciencedirect.com/science/article/pii/S0020025518308016.
[43] J. Zhang and A. C. Sanderson, JADE: Adaptive differential evolution with optional external archive, *IEEE Transactions on Evolutionary Computation*, 13 (2009), 945–958.

Received January 2019; revised November 2019.

E-mail address: smejkto5@fjfi.cvut.cz
E-mail address: jiri.mikyska@fjfi.cvut.cz
E-mail address: jaromir.kukal@fjfi.cvut.cz