Soft Comput (2017) 21:3081–3100
DOI 10.1007/s00500-015-1993-x

METHODOLOGIES AND APPLICATION

Conceptual and numerical comparisons of swarm intelligence optimization algorithms

Haiping Ma[1,2] · **Sengang Ye**[1] · **Dan Simon**[3] · **Minrui Fei**[2]

Published online: 26 December 2015

Abstract Swarm intelligence (SI) optimization algorithms are fast and robust global optimization methods, and have attracted significant attention due to their ability to solve complex optimization problems. The underlying idea behind all SI algorithms is similar, and various SI algorithms differ only in their details. In this paper we discuss the algorithmic equivalence of particle swarm optimization (PSO) and various other newer SI algorithms, including the shuffled frog leaping algorithm (SFLA), the group search optimizer (GSO), the firefly algorithm (FA), artificial bee colony algorithm (ABC) and the gravitational search algorithm (GSA). We find that the original versions of SFLA, GSO, FA, ABC, and GSA, are all algorithmically identical to PSO under certain conditions. We discuss their diverse biological motivations and algorithmic details as typically implemented, and show how their differences enhance the diversity of SI research and application. Then we numerically compare SFLA, GSO, FA, ABC, and GSA, with basic and advanced versions on some continuous benchmark functions and combinatorial knapsack problems. Empirical results show that an advanced version of ABC performs best on the continuous benchmark functions, and advanced versions of SFLA and GSA perform best on the combinatorial knapsack problems. We conclude that although these SI algorithms are conceptually equivalent, their implementation details result in notably different performance levels.

Keywords Evolutionary algorithms · Swarm intelligence · Particle swarm optimization · Conceptual difference · Numerical comparison

Communicated by V. Loia.

✉ Haiping Ma
mahp@usx.edu.cn

[1] Department of Electrical Engineering, Shaoxing University, Shaoxing, Zhejiang, China

[2] Shanghai Key Laboratory of Power Station Automation Technology, School of Mechatronic Engineering and Automation, Shanghai University, Shanghai, China

[3] Department of Electrical Engineering and Computer Science, Cleveland State University, Cleveland, OH, USA

1 Introduction

Population-based optimization methods, such as evolutionary algorithms (EAs) and swarm intelligence (SI) algorithms, are widely used to solve optimization problems and have attracted continually increasing attention in recent years. EAs such as genetic algorithms (GAs) (Reeves and Rowe 2003), evolutionary programming (EP) (Yao et al. 1999), evolution strategies (ES) (Schwefel 1995), and genetic programming (GP) (Koza 1992), are inspired by natural selection. Swarm intelligence (SI) algorithms, such as particle swarm optimization (PSO) (Kennedy and Eberhart 1995) and ant colony optimization (ACO) (Dorigo and Gambardella 1997), are inspired by the collective behavior observed in biological systems. SI algorithms are a class of search methods that are based on the learning process of a collection of individuals, called candidate solutions. Each individual in an SI algorithm performs a set of operations that may include random search, positive or negative feedback, and interactions with other individuals. These operations are typically very simple, but their combination can produce surprising results.

In recent years, many new SI algorithms, including the shuffled frog leaping algorithm (SFLA) (Eusuff and Lansey 2003), the group search optimizer (GSO) (He et al. 2006), the firefly algorithm (FA) (Yang 2009), artificial bee colony algorithm (ABC) (Karaboga and Basturk 2007), the gravitational search algorithm (GSA) (Rashedi et al. 2009), and

others, have been proposed to solve complex optimization problems. Each SI algorithm is motivated by a different biological process. SFLA is motivated by the leaping behavior of frogs, GSO is motivated by the food foraging behavior of animals, FA is motivated by the mutual attraction of fireflies, ABC is motivated by the foraging behavior of bees, and GSA is motivated by the law of gravity.

But all of these SI algorithms have certain features in common, including the sharing of information between individuals to improve fitness. This behavior makes SI algorithms applicable to many different types of optimization problems. SI algorithms have been applied to many optimization problems and have been shown to be effective for unimodal, multimodal, and deceptive optimization, dynamic optimization, constrained optimization, multi-objective optimization, and noisy optimization (Simon 2013). So SI algorithms are becoming increasingly popular optimization tools.

The goal of this paper is to show the similarities and differences between PSO and various new SI algorithms, including SFLA, GSO, FA, ABC, and GSA, in both a notional and experimental way. Because PSO and newer SI algorithms have similarities due to their biological motivation, it is not too surprising that the algorithms are equivalent under special conditions. In this paper, we provide general descriptions of these algorithms and provide conceptual and numerical comparisons.

There have been many papers that compare various SI algorithms, including comparisons of PSO and SFLA (Elbeltagi et al. 2005); PSO and GSA (Davarynejad et al. 2014); PSO and FA (Yang 2011; Parpinelli and Lopes 2011); PSO, BFO, and ABC (Parpinelli et al. 2012); cuckoo search, PSO, DE, and ABC (Civicioglu and Besdok 2013); and others (Zang et al. 2010). Those papers focus on performance differences on benchmarks or applications. In this paper we provide a more extensive comparison by studying similarities and differences between the algorithms, by studying more recently SI algorithms, and including a larger and more recent set of benchmarks. The benchmarks in this paper are the continuous functions from the 2013 IEEE Congress on Evolutionary Computation (Liao and Stuetzle 2013), and a set of classical combinatorial knapsack problems (Abulkalamazad et al. 2014; Bhattacharjee and Sarmah 2014; Freville 2004).

There are many other SI algorithms that we could study, including bat algorithm (BA) (Hasançebi and Carbas 2014), Cuckoo search (CS) (Cobos et al. 2014), fireworks algorithm (FWA) (Tan and Zhu 2010), and teaching-learning-based optimization (TLBO) (Rao et al. 2011). Because of space constraints we restrict our comparison to PSO, SFLA, GSO, FA, ABC, and GSA, and we defer the comparison of other SI algorithms for future work. The SI algorithms in this paper comprise a representative set, not a complete set.

The rest of this paper is organized as follows. Section 2 gives an overview of PSO and the five newer SI algorithms, analyzes their similarities, differences, and unique characteristics. Section 3 presents performance comparisons of the basic and advanced versions of PSO with basic and advanced versions of SFLA, GSO, FA, ABC, and GSA on continuous benchmark functions and combinatorial knapsack problems. Section 4 gives conclusions and directions for future research.

2 Comparisons between swarm intelligence algorithms

This section introduces the basic PSO algorithm and five newer SI algorithms, including SFLA, GSO, FA, ABC, and GSA, and then conceptually analyzes their similarities (Sect. 2.1). This section also compares their biological motivations and algorithmic details (Sect. 2.2).

2.1 Similarities between swarm intelligence algorithms

2.1.1 Particle swarm optimization

PSO, introduced by Kennedy and Eberhart (1995), was motivated by the swarming behavior of birds. PSO consists of a swarm of particles moving through an n-dimensional search space of an optimization problem. Every particle has a position vector x_k that encodes a candidate solution to the optimization problem, and a velocity vector v_k that characterizes its velocity in the search space. Each particle remembers its own previously found best position P_{best}, and the global best position G_{best} of the entire swarm in the current iteration. Information from good solutions spreads throughout the swarm, which encourages the particles to move to better areas of the search space. Each iteration, the particles' velocities are updated based on their current velocities, their previous best positions, and the global best position of the current iteration:

$$\begin{aligned} v_k(s) \leftarrow\ & w v_k(s) + U(0,\phi_1)(P_{best}(s) - x_k(s)) \\ & + U(0,\phi_2)(G_{best}(s) - x_k(s)), \end{aligned} \tag{1}$$

where $v_k(s)$ is the sth dimension of the velocity of the kth particle, w is the inertia weight which determines the contribution of the current velocity to the new velocity, $U(a,\ b)$ is a uniformly distributed random number between a and b, and cognitive constant ϕ_1 and social constant ϕ_2 determine the importance of $P_{best}(s)$ and $G_{best}(s)$, respectively, for updating velocity. The position of the particle is updated as follows:

$$x_k(s) \leftarrow x_k(s) + v_k(s). \tag{2}$$

An outline of the basic PSO algorithm is given in Algorithm I.

Algorithm I – Outline of the basic PSO algorithm, where Equations (1) and (2) are combined

Initialize a random population $\{x_k\}$ for $k \in [1, N]$, and N is the population size
While not (termination criterion)
  For each solution (particle's position) x_k
    Find the current global best solution G_{best}
    Find the particle's previous best solution P_{best}
    For each dimension $s \in [1, n]$
      Update: $x_k(s) \leftarrow x_k(s) + w \cdot v_k(s) + U(0, \phi_1)(P_{best}(s) - x_k(s)) + U(0, \phi_2)(G_{best}(s) - x_k(s))$
    Next dimension
  Next solution
Next generation

Algorithm II – Outline of SFLA, where each frog encodes a candidate solution

Initialize a random population $\{x_i\}$ for $i \in [1, N]$, and N is the population size
While not (termination criterion)
  Randomly divide the population into m sub-populations
  For each sub-population $k = 1$ to m
    Find the global best solution x_g in the entire population
    For $i = 1$ to $i_{\max}$, and $i_{\max}$ is an iteration limit
      Find the best and worst solutions in sub-population k : x_b and x_w
      For each dimension $s \in [1, n]$
        Update: $x_w(s) \leftarrow x_w(s) + r \cdot (x_b(s) - x_w(s))$
        If the update did not improve $x_w(s)$ then
          Update: $x_w(s) \leftarrow x_w(s) + r \cdot (x_g(s) - x_w(s))$
          If the update did not improve $x_w(s)$ then
            $x_w(s) \leftarrow$ randomly-generated variable
          End if
        End if
      Next dimension
    Next iteration
  Next sub-population
Next generation

2.1.2 Shuffled frog leaping algorithm

SFLA was introduced by Eusuff and Lansey (2003), and some variations and applications of SFLA are discussed in Wang and Fang (2011), Rahimi-Vahed and Mirzaei (2008), and Sarkheyli et al. (2015). SFLA is inspired by the foraging behavior of frogs, and consists of a swarm of frogs leaping in the n-dimensional search space of an optimization problem. In SFLA, every frog encodes a candidate solution, and the N frogs are divided into m sub-populations, also called memeplexes. Each sub-population is considered a separate culture, and each sub-population performs a local search algorithm. At the beginning of each generation, the population is shuffled so that each frog is randomly assigned to a new sub-population.

During local search, only the worst frog x_w in each sub-population is updated:

$$x_w(s) \leftarrow x_w(s) + r \cdot (x_b(s) - x_w(s)), \tag{3}$$

where s is the index of the problem dimension, x_b is the best frog in the sub-population, and $r \in [0, 1]$ is a uniformly distributed random number. If (3) does not improve x_w, it is updated again as follows:

$$x_w(s) \leftarrow x_w(s) + r \cdot (x_g(s) - x_w(s)) \tag{4}$$

for $s \in [1, n]$, where x_g is the global best frog from all m sub-populations, and $r \in [0, 1]$ is a random number that is calculated separate for each s. If (4) does not improve

Algorithm III –Outline of the modified PSO algorithm

Initialize a random population $\{x_k\}$
for $k \in [1, N]$, and N is the population size
While not (termination criterion)
  Find the current global worst solution x_w
  Find the current global best solution G_{best}
  Find the previous best solution P_{best}
  For each dimension $s \in [1,n]$ of x_w
    Update:

$$x_w(s) \leftarrow x_w(s) + v_k(s) + U(0,\phi_1)(P_{best}(s) - x_w(s)) + U(0,\phi_2)(G_{best}(s) - x_w(s))$$

  Next dimension
Next generation

Algorithm IV – Outline of the modified SFLA

Initialize a random population $\{x_k\}$
for $k \in [1, N]$, and N is the population size
While not (termination criterion)
  Find the current global worst solution x_w
  Find the current global best solution x_g
  Find the previous best solution x_b
  For each dimension $s \in [1,n]$
    Update:

$$x_w(s) \leftarrow x_w(s) + x_r(s) + r_1 \cdot (x_b(s) - x_w(s)) + r_2 \cdot (x_g(s) - x_w(s))$$

  Next dimension
Next generation

x_w, then x_w is replaced by a randomly generated frog. A description of SFLA is given in Algorithm II.

Suppose that the PSO logic of Algorithm I only updates the global worst solution x_w instead of each solution x_k. Further suppose that the velocity v_k is randomly generated, and the inertia weight is $w = 1$. Algorithm I then becomes the modified PSO of Algorithm III.

Now suppose that in the SFLA logic of Algorithm II, the population is not divided into sub-populations; that is, $m = 1$. Further suppose the best sub-population solution x_b is set to the previous best solution of the individual, and the worst sub-population solution x_w is set to the current worst solution of the entire population. Finally, suppose that $i_{\max} = 1$. Then the SFLA logic of Algorithm II becomes the modified SFLA logic of Algorithm IV, which is equivalent to the modified PSO of Algorithm III. So SFLA with special tuning parameters can be viewed as a type of PSO algorithm, and it follows that these two algorithms perform identically under these conditions.

2.1.3 Group search optimization

GSO was introduced by He et al. (2006), and some variations and applications of GSO are discussed in Shen et al. (2009), Chen et al. (2012), Wang et al. (2012) and Zare et al. (2012). GSO is inspired by the food foraging behavior of land animals, and consists of a swarm of animals foraging in the n-dimensional search space of an optimization problem. Some animals, called producers, focus their efforts on searching for food. Other animals, called joiners or scroungers, focus their efforts on following other animals and exploiting their food-finding success. The third type of animals, called rangers, performs a random walk in their search for resources. In GSO, all animals encode candidate solutions, and the producers are typically the best. Each generation, the producer scans three points in his immediate surroundings for a better function cost value than his current location in search space. This corresponds to local search. The producer is denoted as x_p, and the three scanned points x_z, x_r, and x_l are given by:

$$\begin{aligned} x_z(s) &= x_p(s) + r_1 \cdot l_{\max} \cdot D(\phi_p(s)) \\ x_r(s) &= x_p(s) + r_1 \cdot l_{\max} \cdot D(\phi_p(s) + r_2\theta_{\max}/2) \\ x_l(s) &= x_p(s) + r_1 \cdot l_{\max} \cdot D(\phi_p(s) - r_2\theta_{\max}/2) \end{aligned} \tag{5}$$

for $s \in [1,\ n]$, where r_1 is a zero-mean, unity-variance, normally distributed random number; $r_2 \in [0,\ 1]$ is a uniformly distributed random number; ϕ_p is the heading angle of x_p; $l_{\max}$ is a tuning parameter that defines how far the producer can see; $\theta_{\max}$ is a tuning parameter that defines how far the producer can turn his head; and $D(\cdot)$ is a polar-to-Cartesian coordinate transformation function (He et al. 2006).

Scroungers generally move toward the producer. But they do not move directly toward the producer; instead they move in a sort of zig-zag pattern, which allows them to search for lower cost function values while they move. The movement of a scrounger x_i is modeled as

$$x_i(s) \leftarrow x_i(s) + r \cdot (x_p(s) - x_i(s)), \tag{6}$$

where $r \in [0,\ 1]$ is a uniformly distributed random number.

Rangers randomly travel through the search space looking for areas with low cost functions. The movement of a ranger x_i is modeled as

$$\begin{aligned} \phi_i(s) &\leftarrow \phi_i(s) + \rho \cdot \alpha_{\max} \\ x_i(s) &\leftarrow x_i(s) + \alpha_{\max} \cdot l_{\max} \cdot r_1 \cdot D(\phi_i(s)), \end{aligned} \tag{7}$$

where $\alpha_{\max}$ is a tuning parameter that defines how far a ranger can turn his head; $\rho \in [-1,\ 1]$ is a uniformly distributed random number; and $l_{\max}$ is a tuning parameter that is related to the maximum distance that a ranger can travel in one generation, and is the same as $l_{\max}$ in (5).

A description of GSO is given in Algorithm V. Note that algorithm V specifies that one solution is a producer, about 80 % of the solutions are scroungers, and about 20 % of the solutions are rangers.

Suppose that in the PSO logic of Algorithm I, the current global best solution G_{best} is first updated by the sum of the

Algorithm V – Outline of GSO for a minimizaton problem $f(x)$

Initialize a random population $\{x_k\}$ for $k \in [1, N]$, and N is the population size
While not (termination criterion)
  Find the producer x_p which is the current global best solution
  For each dimension $s \in [1, n]$ of x_p

$$x_z(s) = x_p(s) + r_1 \cdot l_{\max} \cdot D(\phi_p(s))$$
$$x_r(s) = x_p(s) + r_1 \cdot l_{\max} \cdot D(\phi_p(s) + r_2\, \theta_{\max}/2)$$
$$x_l(s) = x_p(s) + r_1 \cdot l_{\max} \cdot D(\phi_p(s) - r_2\, \theta_{\max}/2)$$

  Next dimension
  If $\min\{f(x_z), f(x_r), f(x_l)\} < f(x_p)$ then

$$x_p \leftarrow \arg\min\{f(x_z), f(x_r), f(x_l)\}$$

  Else

$$\phi_p \leftarrow \phi_p + \rho\alpha_{\max}$$

  End if
  For each solution $x_k \neq x_p$
    For each dimension $s \in [1, n]$
      If $r_3 < 0.8$
        Let $x_k(s)$ scrounge: $x_k(s) \leftarrow x_k(s) + r \cdot (x_p(s) - x_k(s))$
      Else
        Let $x_k(s)$ range:

$$\phi_i(s) \leftarrow \phi(s) + \rho\alpha_{\max}$$
$$x_i(s) \leftarrow x_i(s) + \alpha_{\max} l_{\max} r_1 D(\phi_i(s))$$

      End if
    Next dimension
  Next solution
Next generation

previous solution and the new velocity $r_1 \cdot v_{best}$, where v_{best} is the velocity of G_{best}. Then the other solutions are divided into two sub-populations. One is about 80 % of the solutions, for which the constant ϕ_1 that determines the significance of $P_{best}(s)$ is set to 0, and which are updated independently of their previous velocities; that is, the inertia weight $w = 0$. The other sub-population is about 20 % of the solutions, and the constant ϕ_1 which determines the significance of $P_{best}(s)$ and the constant ϕ_2 which determines the significance of $G_{best}(s)$ are both set to 0. In this case, Algorithm I becomes the modified PSO logic of Algorithm VI.

Now suppose that in the GSO logic of Algorithm V, the three children solutions x_z, x_r, and x_l perform the same search strategy as that of x_z, directly use the heading angle ϕ_p instead of the coordinate transformation function $D(\cdot)$, and use the tuning parameter $l_{\max} = 1$ and the parameter $\rho = 0$. The GSO logic of Algorithm V becomes the modified GSO logic of Algorithm VII, which is equivalent to the modified PSO logic of Algorithm VI, where the heading angle ϕ is treated as the velocity v and the quality $\alpha_{\max} r_1$ is treated as the inertia weight w. So GSO can be viewed as a variation of PSO under special conditions, and it follows that these two algorithms perform identically under these conditions.

2.1.4 Firefly algorithm

FA was introduced by Yang (2009), and some variations and applications of FA are discussed in Fister et al. (2013). FA is inspired by the mutual attraction of fireflies, which is based on perceived brightness, which decreases exponentially with distance. A firefly is attracted only to those fireflies that are brighter than itself. In FA, every firefly encodes a candidate solution, and the population consists of N fireflies. Each firefly x_i compares its brightness with every other firefly x_j, one at a time. If x_j is brighter than x_i, then x_i will move through the search space in a direction that includes both a

Algorithm VI – Outline of a modified PSO

Initialize a random population $\{x_k\}$ for $k \in [1, N]$, and N is the population size
While not (termination criterion)
  Find the current global best solution G_{best}
  For each dimension $s \in [1,n]$ of G_{best}
    $G'_{best}(s) \leftarrow G_{best}(s) + r_1 \cdot v_{best}(s)$
  Next dimension
  If $f(G'_{best}) < f(G_{best})$ then
    $G_{best} \leftarrow G'_{best}$
  End if
  For other solutions $x_k \neq G_{best}$
    For each dimension $s \in [1,n]$
      If $r_3 < 0.8$
        Update:
        $x_k(s) \leftarrow x_k(s) + U(0,\phi_2)(G_{best}(s) - x_k(s))$
      else
        Update:
        $x_k(s) \leftarrow x_k(s) + w \cdot v_k(s)$
      Next dimension
    Next solution
Next generation

Algorithm VII – Outline of a modified GSO

Initialize a random population $\{x_k\}$ for $k \in [1, N]$, and N is the population size
While not (termination criterion)
  Find the producer x_p which is the current global best solution
  For each dimension $s \in [1,n]$ of x_p
    $x_z(s) = x_p(s) + r_1 \cdot \phi_p(s)$
  Next dimension
  If $f(x_z) < f(x_p)$ then
    $x_p \leftarrow x_z$
  End if
  For each $x_k \neq x_p$
    For each dimension $s \in [1,n]$
      If $r_2 < 0.8$
        Let $x_k(s)$ scrounge:
        $x_k(s) \leftarrow x_k(s) + r \cdot (x_p(s) - x_k(s))$
      Else
        Let $x_k(s)$ range:
        $x_k(s) \leftarrow x_k(s) + \alpha_{\max} r_1 \phi_k(s)$
      End if
    Next dimension
  Next solution
Next generation

component that is random, and a component that is directed toward x_j. The random component is denoted by the quantity αv_k, which is usually relatively small due to the small value of α. The directed component is denoted by the quantity $\beta_0 e^{-\gamma_i r_{ij}^2}(x_j - x_i)$, and its magnitude is an exponential function of the distance r_{ij} between x_j and x_i. Typical tuning parameters of FA are as follows:

Algorithm VIII – Outline of FA for the minimization of $f(x)$

Initialize a random population $\{x_k\}$ for $k \in [1, N]$, and N is the population size
While not (termination criterion)
  For each solution x_k
    For each solution $x_j \neq x_k$
      If $f(x_j) < f(x_k)$
        For each dimension $s \in [1,n]$
          If the uniformly distributed random number $\rho < 1/2$
            $v_k(s) \leftarrow r \cdot (u(s) - x_k(s))$
          Else
            $v_k(s) \leftarrow r \cdot (x_k(s) - l(s))$
          End if
        Next dimension
        $r_{kj} \leftarrow$ distance between x_k and x_j
        Update: $x_k \leftarrow x_k + \beta_0 e^{-\gamma_i r_{kj}^2}(x_j - x_k) + \alpha v_k$
      End if
    Next solution
  Next solution
Next generation

$$\gamma_i = \frac{\gamma_0}{\max_j \|x_i - x_j\|_2}, \quad \text{where } \gamma_0 = 0.8$$
$$\alpha = 0.01 \tag{8}$$
$$\beta_0 = 1$$

A description of FA is given in Algorithm VIII, where l and u are the lower and upper bounds of the search space, respectively. One thing that we notice from Algorithm VIII is that the best candidate solution in the population is never updated. We might be able to improve the algorithm's performance if we periodically update the best candidate solution.

Suppose that in the PSO logic of Algorithm I, the best candidate solution in the population is never updated, and each particle's velocity is independent of its previous velocity; that is, the inertia weight $w = 0$. Further suppose that half of the solutions are determined solely by the current global best solution G_{best}, and the other half of the solutions are determined solely by their previous best solution P_{best}. In this case, the PSO logic of Algorithm I becomes the modified PSO logic of Algorithm IX.

Now suppose that in the FA of Algorithm VIII, the lower bound l and the upper bound u of the search space are replaced with the current global best solution G_{best} and the previous individual best solution P_{best}. Further suppose the parameter $\gamma \to \infty$, which denotes that fireflies are not attracted to each other, and which corresponds to random flight and random search. In this case, the FA of Algorithm

Algorithm IX – Outline of a modified PSO

Initialize a random population $\{x_k\}$ for $k \in [1, N]$, and
N is the population size
While not (termination criterion)
  For each solution (particle's position) x_k
    For each solution $x_j \neq x_k$
      If $f(x_j) < f(x_k)$
        Find the current global best solution G_{best}
        Find the previous best solution P_{best}
        For each dimension $s \in [1, n]$
          If the uniformly distributed random number $\rho < 1/2$
            $v_k(s) \leftarrow U(0, \phi_1)(P_{best}(s) - x_k(s))$
          Else
            $v_k(s) \leftarrow U(0, \phi_2)(G_{best}(s) - x_k(s))$
          End if
          Update: $x_k \leftarrow x_k + v_k$
        Next dimension
      End if
    Next solution
  Next solution
Next generation

VIII is equivalent to the modified PSO of Algorithm IX. So FA can be viewed as a variation of PSO under special conditions, and it follows that these two algorithms perform identically under these special conditions.

2.1.5 Gravitational search algorithm

GSA was introduced by Rashedi et al. (2009), and some variations and applications of GSA are discussed in Dowlatshahi et al. (2014), Jiang et al. (2014) and Rashedi et al. (2011). GSA is inspired by the law of gravity; each particle attracts every other particle due to gravity. The gravitational force between particles is proportional to the product of their masses, and inversely proportional to the square of the distance between them. In GSA, each particle has four characteristics: position, inertial mass, active gravitational mass, and passive gravitational mass. The position of each particle encodes a candidate solution of the optimization problem, and its gravitational and inertial masses are determined by the fitness function. All particles attract each other due to gravity, which causes a global movement toward the particles with heavier masses (better fitness values). Particles thus cooperate with each other using a direct form of communication through gravity. Heavy particles (that is, more fit particles) move more slowly than lighter ones. In other words, the algorithm is tuned by adjusting the gravitational and inertial masses. A description of GSA is given in Algorithm X, where g is the gravitational constant, M is the normalized fitness, R and F are the distance and force between particles, respectively, a and v are the acceleration and velocity, respectively, r and r_i are uniformly distributed random numbers, t and $t_{\max}$ are the generation number and the generation limit, respectively, and ε is a small positive constant.

In the GSA of Algorithm X, the gradual reduction of the gravitational constant g reduces the exploration component of the algorithm as time progresses. The fitness values are normalized so that the worst particle has gravitational mass $m_i = 0$ and the best particle has gravitational mass $m_i = 1$. These masses are normalized to $\{M_i\}$ values that sum to 1. For each pair of particles, the attractive force is calculated with a magnitude that is proportional to their masses and inversely proportional to the distances between them. A random combination of the forces results in the acceleration vector of each particle. The acceleration is used to update the velocity and position of each particle. A more compact representation of Algorithm X, in which distance, force, acceleration, and position are combined, is shown in Algorithm XI.

Suppose that in the PSO logic of Algorithm I the fitness values of all solutions are normalized, and the constant ϕ_1 which determines the significance of P_{best} is set to 0. Further suppose that the update term of the current global best position includes the coefficient $M_{best}/(\|G_{best}(s) - x_k(s)\|_2 + \varepsilon)$, where M_{best} is the normalized fitness of G_{best}, and ε is a small positive constant to prevent division by zero. In this case, the PSO logic of Algorithm I becomes the modified PSO logic of Algorithm XII.

Now suppose that the GSA of Algorithm XI calculates acceleration based only on the current global best solution G_{best}, instead of using all of the solutions in the entire population. The GSA of Algorithm XI then becomes the modified GSA of Algorithm XIII. We find that if in Algorithm XII the inertia weight w is set to a random number, and if in Algorithm XIII the gravitational constant g is set to 1, then the modified GSA is equivalent to the modified PSO. So GSA can be viewed as a variation of PSO under special conditions, and it follows that these two algorithms perform identically under these special conditions.

2.1.6 Artificial bee colony algorithm

ABC was introduced by Karaboga (2005) and some variations and applications of ABC are discussed in Dervis et al. (2014) and Dervis and Bahriye (2007). ABC is inspired by the behavior of bees as they search for an optimal food source. The location of a food source is analogous to a location in the search space of an optimization problem. The amount of nectar at a location is analogous to the fitness of a candidate solution. In ABC, there are three different types of bees: forager bees, onlooker bees, and scout bees. First, forager bees travel back and forth between a food source and their hive. Each forager is associated with a specific location, and remembers that location as it travels back and forth between the hive. When a forager takes its nectar to the hive, it returns to its food source, but it also engages in local exploration as it searches

Algorithm X – Outline of GSA for the optimization of $f(x)$

Initialize a random population $\{x_k\}$ for $k \in [1, N]$, and N is the population size

While not (termination criterion)

Gravitational constant $g \leftarrow g_0 \exp(-\alpha t / t_{\max})$

Fitness: $m_k \leftarrow \dfrac{f(x_k) - \text{worst}_i(f(x_i))}{\text{best}_i(f(x_i)) - \text{worst}_i(f(x_i))} \in [0,1]$ for $k \in [1, N]$

Normalized fitness: $M_k \leftarrow \dfrac{m_k}{\sum_{i=1}^{N} m_i}$ for $k \in [1, N]$

For each solution x_k, $k \in [1, N]$

For each dimension $s \in [1, n]$

Distance: $R_{ki}(s) \leftarrow \left\| x_i(s) - x_k(s) \right\|_2$ for $i \in [1, N]$

Force: $F_{ki}(s) \leftarrow \dfrac{g M_k M_i}{R_{ki}(s) + \varepsilon}(x_i(s) - x_k(s))$ for $i \in [1, N]$

Acceleration: $a_k(s) \leftarrow \dfrac{1}{M_k} \sum_{i=1, i \neq k}^{N} r_i F_{ki}(s)$

Velocity: $v_k(s) \leftarrow r v_k(s) + a_k(s)$

Position: $x_k(s) \leftarrow x_k(s) + v_k(s)$

Next dimension

Next solution

Increment generation number: $t \leftarrow t + 1$

Next generation

in the nearby vicinity for a better source. Second, onlooker bees are not associated with any particular food source, but they observe the amount of nectar that is returned by the foragers (that is, the fitness of each forager's location in search space), and use that information to decide where to search for nectar. The onlookers' search location is decided probabilistically based on their observations of the foragers. Third, scout bees are explorers, and are also not associated with any particular food source. If a scout sees that a forager has stagnated and is not progressively increasing the amount of nectar that it returns to the hive, the scout randomly searches for a new nectar source in the search space. Stagnation is indicated when the explorer fails to increase the amount of nectar it brings to the hive after a certain number of trips.

A description of ABC is given in Algorithm XIV, where P_f is the forager population size and P_0 is the onlooker population size. Note that the total population size is $N = P_0 + P_f$. L is a positive integer, which denotes the stagnation limit, $r \in [-1, 1]$ is a uniformly distributed random number, $T(\cdot)$ is a forager trial counter that keeps track of the number of consecutive unsuccessful modifications of each forager.

Suppose that in the PSO logic of Algorithm I, the solutions are divided into two sub-populations. For one sub-population, the constant ϕ_1 that determines the significance of $P_{best}(s)$ is set to 0, and which are updated independently of their previous velocities; that is, the inertia weight is $w = 0$. For another sub-population, the constant ϕ_2 which determines the significance of $G_{best}(s)$ are set to 0, the inertia weight is $w = 0$, and the current solution x_k is replaced by the current global best solution G_{best}. In addition, randomly selected one dimension in candidate solution is updated instead of all dimensions in each iteration. In this case, Algorithm I becomes the modified PSO logic of Algorithm XV.

Now suppose that in the ABC method of Algorithm XIV we do not set the forager trial counters; that is, the performance of scout bees is not considered. The ABC method of Algorithm XIV then becomes the modified ABC method of Algorithm XVI. We find that if in the first sub-population in

Algorithm XI – Outline of compact version of GSA for the optimization of $f(x)$

Initialize a random population $\{x_k\}$ for $k \in [1, N]$, and N is the population size

While not (termination criterion)

Gravitational constant $g \leftarrow g_0 \exp(-\alpha t / t_{\max})$

Fitness: $m_k \leftarrow \frac{f(x_k) - \text{worst}_i(f(x_i))}{\text{best}_i(f(x_i)) - \text{worst}_i(f(x_i))} \in [0,1]$ for $k \in [1, N]$

Normalized fitness: $M_k \leftarrow \frac{m_k}{\sum_{i=1}^{N} m_i}$ for $k \in [1, N]$

For each individual x_k, $k \in [1, N]$

For each dimension $s \in [1, n]$

Update: $x_k(s) \leftarrow x_k(s) + r v_k(s) + \sum_{i=1, i \neq k}^{N} r_i \frac{g M_i}{\|x_i(s) - x_k(s)\|_2 + \varepsilon} (x_i(s) - x_k(s))$

Next dimension

Next solution

Increment generation number: $t \leftarrow t + 1$

Next generation

Algorithm XII – Outline of a modified PSO algorithm

Initialize a random population $\{x_k\}$ for $k \in [1, N]$, and N is the population size

While not (termination criterion)

Fitness: $m_k \leftarrow \frac{f(x_k) - \text{worst}_i(f(x_i))}{\text{best}_i(f(x_i)) - \text{worst}_i(f(x_i))} \in [0,1]$

for $k \in [1, N]$

Normalized fitness: $M_k \leftarrow \frac{m_k}{\sum_{i=1}^{N} m_i}$ for $k \in [1, N]$

For each solution (particle's position) x_k

Find the current global best solution G_{best} and its fitness M_{best}

For each dimension $s \in [1, n]$

Update:

$x_k(s) \leftarrow x_k(s) + w \cdot v_k(s)$

$+U(0, \phi_2) \frac{M_{best}}{\|G_{best}(s) - x_k(s)\|_2 + \varepsilon} (G_{best}(s) - x_k(s))$

Next dimension

Next solution

Next generation

Algorithm XIII – Outline of a modified GSO

Initialize a random population $\{x_k\}$ for $k \in [1, N]$, and N is the population size

While not (termination criterion)

Gravitational constant $g \leftarrow g_0 \exp(-\alpha t / t_{\max})$

Fitness: $m_k \leftarrow \frac{f(x_k) - \text{worst}_i(f(x_i))}{\text{best}_i(f(x_i)) - \text{worst}_i(f(x_i))} \in [0,1]$

for $k \in [1, N]$

Normalized fitness: $M_k \leftarrow \frac{m_k}{\sum_{i=1}^{N} m_i}$ for $k \in [1, N]$

For each solution x_k, $k \in [1, N]$

Find the current global best solution G_{best} and its fitness M_{best}

For each dimension $s \in [1, n]$

Update:

$x_k(s) \leftarrow x_k(s) + r_1 v_k(s)$

$+r_2 \frac{g M_{best}}{\|G_{best}(s) - x_k(s)\|_2 + \varepsilon} (G_{best}(s) - x_k(s))$

Next dimension

Next solution

Increment generation number: $t \leftarrow t + 1$

Next generation

Algorithm XV the current global best solution G_{best} is set to a random number, and in the second sub-population the current global best solution G_{best} is chosen using roulette-wheel selection, and the particle's previous best solution P_{best} is set to a random number, then the modified ABC method is equivalent to the modified PSO algorithm. So ABC can be viewed as a variation of PSO under special conditions, and it follows that these two algorithms perform identically under these special conditions.

2.1.7 Summary of swarm intelligence algorithm comparisons

It has seen in the above sections that several new SI algorithms, including SFLA, GSO, FA, ABC and GSA, are conceptually similar to PSO. Under special conditions, these

Algorithm XIV – Outline of ABC for the optimization of $f(x)$, where each forager and onlooker encodes a candidate solution

Initialize a random population of foragers $\{x_k\}$ for $k \in [1, P_f]$, and onlooker $\{v_i\}$ for $i \in [1, P_0]$
While not (termination criterion)
  Forager bees:
  For each forager x_k
    $i \leftarrow$ random integer $\in [1, N]$ such that $i \neq k$; $s \leftarrow$ random integer $\in [1, n]$
    $v_k(s) \leftarrow x_k(s) + r \cdot (x_k(s) - x_i(s))$
    If $f(v_k)$ is better than $f(x_k)$ then
      $x_k \leftarrow v_k$; $T(x_k) \leftarrow 0$
    Else
      $T(x_k) \leftarrow T(x_k) + 1$
    End if
  Next forager
  Onlooker bees:
  For each onlooker v_i
    Select a forager x_k, where $\Pr(x_k) \propto \text{fitness}(x_k)$ for $k \in [1, P_f]$
    $j \leftarrow$ random integer $\in [1, P_f]$ such that $j \neq k$; $s \leftarrow$ random integer $\in [1, n]$
    $v_i(s) \leftarrow x_k(s) + r \cdot (x_k(s) - x_j(s))$
    If $f(v_i)$ is better than $f(x_k)$ then
      $x_k \leftarrow v_i$; $T(x_k) \leftarrow 0$
    Else
      $T(x_k) \leftarrow T(x_k) + 1$
    End if
  Next onlooker
  Scout bees:
  For each forager x_k, $k \in [1, P_f]$
    If $T(x_k) > L$ then
      $x_k \leftarrow$ randomly-generated individual
      $T(x_k) \leftarrow 0$
    End if
  Next forager
Next generation

algorithms are equivalent to PSO variations. All of these algorithms have certain features in common and can be viewed as variations on the same themes, including inertia, influence by society, and influence by neighbors. Since they have so many similarities, it is easy to see why they have similar performance in many real-world optimization problems. We note that there are many other new SI algorithms, including glowworm swarm optimization (GSO) (Krishnanand and Ghose 2009), grey wolf optimizer (GWO) (Mirjalili and Lewis 2014), and others, but the study of their similarities and differences is deferred for future research.

2.2 Conceptual differences between swarm intelligence algorithms

The identical functionality of different SI algorithms discussed above occurs only under special conditions. Each SI algorithm still has its own particular features and operations that can give it flexibility that other SI algorithms may not have. In this subsection, we point out some differences between various SI algorithms.

2.2.1 Biological motivation and its effect on future research

Differences between SI algorithms result from their unique biological motivations. PSO is based on the swarming behavior of birds, SFLA is based on the leaping behavior of frogs, GSO is based on the food foraging behavior of land-based animals, FA is based on the attraction of fireflies to one another, ABC is based on the foraging behavior of bees, and GSA is based on the law of gravity. It is therefore useful to retain the distinction between these SI algorithms because they are based on different phenomena, and those distinct phenomena can be used to introduce helpful algorithmic variations for improved performance.

Retaining the swarm foundation of PSO stimulates research toward the incorporation of social behavior from animals, which can enrich and extend the study of PSO. Some

Algorithm XV – Outline of a modified PSO algorithm

Initialize a random sub-population $\{x_k\}$ for $k \in [1, P_f]$, and another sub-population $\{v_i\}$ for $i \in [1, P_0]$
While not (termination criterion)
Sub-population 1:
For each solution (particle's position) x_k
Find the current global best solution G_{best}
$s \leftarrow$ random integer $\in [1, n]$
Update:
$v_k(s) \leftarrow x_k(s) + U(0, \phi_2)(G_{best}(s) - x_k(s))$
If $f(v_k)$ is better than $f(x_k)$ then
$x_k \leftarrow v_k$
End if
Next solution
Sub-population 2:
For each solution (particle's position) v_i
Find the current global best solution G_{best}
Find the particle's previous best solution P_{best}
$s \leftarrow$ random integer $\in [1, n]$
Update:
$v_i(s) \leftarrow G_{best}(s) + U(0, \phi_1)(P_{best}(s) - G_{best}(s))$
If $f(v_i)$ is better than $f(G_{best})$ then
$G_{best} \leftarrow v_i$
End if
Next solution
Next generation

Algorithm XVI – Outline of a modified ABC algorithm

Initialize a random population of foragers $\{x_k\}$ for $k \in [1, P_f]$, and onlooker $\{v_i\}$ for $i \in [1, P_0]$
While not (termination criterion)
Forager bees:
For each forager x_k
$i \leftarrow$ random integer $\in [1, N]$ such that $i \neq k$
$s \leftarrow$ random integer $\in [1, n]$
$v_k(s) \leftarrow x_k(s) + r \cdot (x_k(s) - x_i(s))$
If $f(v_k)$ is better than $f(x_k)$ then
$x_k \leftarrow v_k$
End if
Next forager
Onlooker bees:
For each onlooker v_i
Select a forager x_k,
where $\Pr(x_k) \propto \text{fitness}(x_k)$ for $k \in [1, P_f]$
$j \leftarrow$ random integer $\in [1, P_f]$ such that $j \neq k$
$s \leftarrow$ random integer $\in [1, n]$
$v_i(s) \leftarrow x_k(s) + r \cdot (x_k(s) - x_j(s))$
If $f(v_i)$ is better than $f(x_k)$ then
$x_k \leftarrow v_i$
End if
Next onlooker
Next generation

of these behaviors include avoiding predators, seeking food, and seeking to travel more quickly. Retaining SFLA as a separate algorithm stimulates research toward the incorporation of frog behavior, including complex shuffling behavior and new leaping rules. Retaining GSO as a separate algorithm stimulates research toward the incorporation of foraging behavior from land-based animals, including additional food scanning mechanisms, scrounging strategies, and random search strategies. Retaining FA as a separate algorithm stimulates the research toward the incorporation of firefly behavior, including the dispersion of individuals, consideration of light intensity, and other atmospheric considerations. Retaining ABC as a separate algorithm stimulates research toward the incorporation of bee behavior, including the profitability of a food source, the distance and direction from the nest, and other foraging models. Retaining GSA as a separate algorithm stimulates research toward the incorporation of additional characteristics related to gravity, including active gravitational force, passive gravitational force, and inertia.

2.2.2 Algorithmic differences

Differences in the performance levels of various SI algorithms arise because of differences in the details of these algorithms. Although we have shown that SI algorithms are equivalent under certain conditions, they can operate quite differently as typically implemented. For example, the basic PSO algorithm creates children by updating all solutions based on the current global best solution and the previous best solution. SFLA creates children by updating the worst sub-population solutions based on the current global best solution and the best sub-population solution. GSO creates children by updating each solution based on different search strategies. ABC creates children by updating each solution based on randomly weighted differences of the current solution. GSA creates children by updating each solution based on the same search strategy. FA creates children by updating all solutions except the best one, which is never updated.

Unifying various SI algorithms, as done in this paper, is instructive, but recognizing their individual characteristics and distinctions enables interesting mathematical and theoretical studies, and can provide valuable tools for practical problems. The no-free-lunch theorem (Wolpert and Macready 1997) says that if an algorithm achieves superior results on certain problems, it must pay for that performance with inferior results on other problems. So the existence of various SI algorithms provides an attractive choice of alternate optimization methods. The differences between the algorithms provide the possibility of application to a variety of problems, and for a variety of contributions to the SI literature.

Table 1 summarizes some characteristics of the six algorithms that we consider. The row labeled "Search Domain of Original Formulation" indicates whether the algorithm was originally developed for discrete or continuous search domains. This characteristic is not always obvious. For exam-

Table 1 Summary of the differences among the six SI algorithms that we study

	PSO	SFLA	GSO	FA	ABC	GSA
Year introduced	1995	2003	2006	2009	2005	2009
Search domain of original formulation	Continuous	Discrete	Continuous	Continuous	Continuous	Continuous
Convergence	Slow	Fast	Slow	Fast	Slow	Slow
Best application of original formulation	Most optimization problems	Combinatorial optimization problems	Multimodal optimization problems	Multimodal optimization problems	Most optimization problems	Most optimization problems

ple, SFLA was originally applied to discrete search spaces, but the algorithm is more naturally suited for continuous search spaces, and it is usually applied that way. It should not be assumed that the various SI algorithms suit only the type of search domain in the table. In fact, they have all been modified to apply to both discrete and continuous search domains, and may even perform better a search domain type that is different than the original search domain type.

The "Convergence" row in Table 1 indicates whether the algorithms have fast or slow convergence, in general. This characteristic is relatively clear-cut; the original versions of SFLA and FA have fast convergence, and the original versions of PSO, GSA, ABC and GSO have slow convergence. However, many variants of PSO, GSA, ABC, and GSO exhibit greatly improved convergence abilities. Note that convergence ability is quantified by empirical evidence, but there is little theoretical evidence to support the convergence characterizations in Table 1.

The "Best Application of Original Formulation" row in Table 1 indicates the type of problem for which the algorithm was initially developed. SFLA was initially applied to combinatorial optimization problems and obtained good performance, but modified versions of SFLA have applied to all types of search domains. The original versions of GSO and FA show good performance for multimodal optimization problems, and the original versions of PSO, ABC and GSA show good performance for the most types of optimization problems.

3 Experimental results

This section investigates the performance of the six SI algorithms considered in this paper, including SFLA, GSO, FA, ABC, GSA, and PSO, along with advanced versions. The advanced versions include modified SFLA (MSFLA) (Emad et al. 2007), self-adaptive group search optimizer with elitist strategy (SEGSO) (Zheng et al. 2014), variable step size firefly algorithm (VSSFA) (Yu et al. 2015), modified artificial bee colony (MABC) (Bahriye and Dervis 2012), GSA with chaotic local search approach (CGSA2) (Gao et al. 2014), PSO with linearly varying inertia weight (LPSO) (Shi and Eberhart 1998), and PSO with constriction factor (CPSO) (Clerc and Kennedy 2002). We select these advanced versions because the literature shows that they generally provide better performance than the basic algorithms. Section 3.1 compares performance on the continuous benchmark functions from the 2013 IEEE Congress on Evolutionary Computation, and Sect. 3.2 compares performance on a set of combinatorial knapsack problems.

3.1 Continuous functions

This subsection compares the performance of SFLA, GSO, FA, ABC, GSA, PSO, and their advanced versions on a set of continuous benchmark functions from the 2013 IEEE Congress on Evolutionary Computation (Liao and Stuetzle 2013). These functions are briefly summarized in Table 2, and the parameters used in this subsection and the corresponding references are shown in Table 3. Note that we did not optimize the parameter values of the algorithms, but instead we used the parameter values that are given in the references in Table 3. Each algorithm has a population size of 200, and each function is evaluated in 50 dimensions with a function evaluation limit of 500,000. All algorithms terminate after the maximum number of function evaluations, or if the objective function value falls below 10^{-8}. All results in this section are computed from 25 independent simulations. Results are shown in Tables 4 and 5.

According to Tables 4 and 5, MABC performs best on 10 functions (F2, F6, F7, F9, F10, F12, F14, F18, F25, and F28), VSSFA performs best on 5 functions (F3, F16, F20, F21, and F27), CGSA2 performs best on 4 functions (F1, F4, F11, and F19), FA performs best 4 functions (F13, F15, F17, and F22), ABC performs best on 2 functions (F5 and F23), MSFLA performs best on 2 functions (F8 and F24), and GSA performs best on function F26. These results indicate that MABC is the most effective, VSSFA is the second most effective, and CGSA2 and FA are the third most effective for the continuous benchmark functions that we studied.

Table 2 2013 CEC benchmark functions, where the search domain of all functions is $-100 \leq x_i \leq 100$

		Function name	Function minimum
Unimodal functions	F1	Sphere function	−1400
	F2	Rotated high conditioned elliptic function	−1300
	F3	Rotated Bent Cigar function	−1200
	F4	Rotated discus function	−1100
	F5	Different powers function	−1000
Basic multimodal functions	F6	Rotated Rosenbrock function	−900
	F7	Rotated Schaffer F7 function	−800
	F8	Rotated Ackley function	−700
	F9	Rotated Weierstrass function	−600
	F10	Rotated Griewank function	−500
	F11	Rastrigin function	−400
	F12	Rotated Rastrigin function	−300
	F13	Discontinuous rotated Rastrigin function	−200
	F14	Schwefel function	−100
	F15	Rotated Schwefel function	100
	F16	Rotated Katsuura function	200
	F17	Lunacek Bi_Rastrigin function	300
	F18	Rotated Lunacek Bi_Rastrigin function	400
	F19	Expanded Griewank plus Rosenbrock function	500
	F20	Expanded Schaffer F6 function	600
Composition multimodal functions	F21	Composition function 1 ($n = 5$, rotated)	700
	F22	Composition function 2 ($n = 3$, unrotated)	800
	F23	Composition function 3 ($n = 3$, rotated)	900
	F24	Composition function 4 ($n = 3$, rotated)	1000
	F25	Composition function 5 ($n = 3$, rotated)	1100
	F26	Composition function 6 ($n = 5$, rotated)	1200
	F27	Composition function 7 ($n = 5$, rotated)	1300
	F28	Composition function 8 ($n = 5$, rotated)	1400

More details about these functions can be found in Liao and Stuetzle (2013)

Furthermore, we find that for most benchmark functions, the performances of the advanced versions of SI algorithms are better than their corresponding basic versions, which indicates that the modifications of the algorithms can improve the optimization performance.

Next we briefly consider the types of functions for which the various algorithms are best-suited. Tables 4 and 5 show that for the unimodal functions (F1–F5), ABC and its advanced version perform best on two functions, and GSA and its advanced version perform best on two functions, and so they are the most effective algorithms. For the basic multimodal functions (F6–F20) and composition multimodal functions (F21–F28), ABC and its advanced version perform best on ten functions and are the most effective algorithms, and FA and its advanced version perform best on 8 functions and are thus the second most effective algorithms. This is consistent with the observation in Table 1 about the best application of the original formulation of the SI algorithms.

The average computing times of the algorithms are shown in the last row of Tables 4 and 5. Here MATLAB® is used as the programming language, and the computer is a 2.40 GHz Intel Pentium® 4 CPU with 4 GB of memory. We find that all algorithms can be ranked from fastest to slowest as follows: FA, VSSFA, SFLA, MSFLA, ABC, MABC, PSO, GSA, LPSO, CPSO, CGSA2, GSO, and SEGSO.

In order to further compare the performance of SI algorithms, we perform a Holm multiple comparison test while considering PSO as the control method, which acts as the base algorithm for comparison. The Holm multiple comparison test is a nonparametric statistical test that obtains a probability (p value) that determines the degree of difference between a control algorithm and a set of alternative algorithms, assuming that the algorithms have statistically significant differences as a whole (Derrac et al. 2011). To quantify whether a set of algorithms shows a statistically significant difference as a whole, we first apply Friedman's test with a significance

Table 3 Parameter settings of SFLA, GSO, FA, ABC, GSA, PSO, and their advanced versions

	Parameters	Values		Parameters	Values
SFLA, MSFLA (Eusuff and Lansey 2003; Emad et al. 2007)	Number of memeplexes	$m = 20$	GSO, SEGSO (He et al. 2006; Zheng et al. 2014)	Initial heading angle of each solution	$\phi_i = \pi/4$
	Number of candidate solution in each memeplex	$n_0 = 10$		Maximum pursuit angle	$\theta_{\max} = \pi/(round\sqrt{N+1})^2$
	Iteration limit	$i_{\max} = 10$		Maximum turning angle	$\alpha_{\max} = \theta_{\max}/2$
	Acceleration factor (only for MSFLA)	$C = 1.6$		Turning parameter	$l_{\max} = \|u - l\|_2$
ABC, MABC (Karaboga 2005; Bahriye and Dervis 2012)	Forager population size	$P_f = N/2$	GSA, CGSA2 (Rashedi et al. 2009; Gao et al. 2014)	Gravitational constant	$g_0 = 100$
	Onlooker population size	$P_0 = N/2$		Decay rate	$\alpha = 20$
	Stagnation limit	$L = Nn/2$		Chaotic logistic map coefficient (only for CGSA2)	$\mu = 4$
	Modification rate (only for MABC)	$MR = 0.4$		Shrinking parameter (only for CGSA2)	$\rho = 0.978$
	Scaling factor (only for MABC)	$SF = 1$			
FA, VSSFA (Yang 2009; Yu et al. 2015)	Tuning parameter	$\gamma_0 = 0.8$	PSO (Kennedy and Eberhart 1995)	Inertia weight	$w = 0.8$
	Tuning parameter	$\alpha = 0.01$		Cognitive constant	$\phi_1 = 1.0$
	Tuning parameter	$\beta_0 = 1$		Social constant	$\phi_2 = 1.0$
LPSO (Shi and Eberhart 1998)	Fixed initial inertia weight	$w_{init} = 0.2$	CPSO (Clerc and Kennedy 2002)	Constriction coefficient	$K = 2/\lvert 2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\rvert \varphi = 4.1$
	Inertia weight slope	$m = -2.5 \times 10^{-4}$		The other parameters of LPSO and CPSO are the same as those of PSO	
	Nonlinear modulation index	$n = 1$			

level $\alpha = 0.05$ to the mean error rankings. If the test rejects the null hypothesis that all of the algorithms perform similarly, we compare the control method with the remaining algorithms according to their rankings. Additional details about the Holm multiple comparison test can be found in Derrac et al. (2011).

Table 6 shows the Friedman ranks of all the SI algorithms for the 2013 CEC benchmark functions. We obtain a Friedman statistic of 130.45 and a corresponding p value of 0.00012. Because the p value is smaller than 0.05, the results strongly indicate statistically significant performance differences among the algorithms.

Table 7 shows the results of a Holm multiple comparison test. It seems from Table 7 that for the 2013 CEC continuous benchmark functions, MABC is the best with an average rank of 3.62, VSSFA is the second best with an average rank of 4.14, and CGSA2 is the third best with an average rank of 4.57. These results are consistent with those shown in Tables 4 and 5. Furthermore, Table 7 shows statistically significant differences between PSO and all other algorithms except SFLA and GSO, as indicated by p values smaller than 0.05. The larger p values for SFLA and GSO, which are 0.08121 and 0.08902, respectively, indicate that although SFLA and GSO obtain better performance than PSO, the differences are not statistically significant.

We do not want to use the above results to draw broad conclusions. First, for SI algorithms, different tuning values might result in significant performance changes. In general, it can be difficult to determine optimum tuning parameters. A small change in a tuning value could change the performance

Table 4 Comparison of the best error values of the 2013 CEC continuous benchmark functions with $D = 50$ for SFLA, GSO, FA, and their advanced versions

Function	SFLA	MSFLA	GSO	SEGSO	FA	VSSFA
F1	2.37E−10 ± 1.33E−11	6.15E−16 ± 2.23E−17	8.36E−07 ± 4.78E−08	6.55E−10 ± 9.58E−11	7.25E−22 ± 1.24E−23	1.60E−22 ± 6.47E−23
F2	1.26E+03 ± 7.21E+02	3.21E+02 ± 1.14E+01	9.68E+06 ± 8.50E+05	3.24E+05 ± 6.64E+04	4.29E+02 ± 2.31E+01	8.13E+01 ± 4.84E+00
F3	3.45E+01 ± 5.29E+00	7.68E+01 ± 2.58E+00	4.32E+01 ± 7.24E+00	4.19E+02 ± 3.72E+01	1.14E+01 ± 7.54E+00	**3.58E+00 ± 1.35E+00**
F4	4.22E+03 ± 6.47E+02	1.25E+02 ± 4.46E+01	1.25E+03 ± 6.31E+02	7.54E+01 ± 5.24E+00	5.12E−04 ± 6.98E−05	4.46E−04 ± 1.14E−05
F5	6.31E−02 ± 1.58E−03	4.46E−01 ± 2.89E−02	7.68E−02 ± 7.48E−03	3.21E−02 ± 1.17E−03	6.38E−15 ± 1.25E−16	2.25E−16 ± 5.56E–17
F6	8.66E+02 ± 2.33E+01	9.01E+02 ± 3.47E+01	3.11E+00 ± 5.32E−01	5.04E+00 ± 1.60E+00	7.42E−02 ± 3.74E−03	7.76E−03 ± 7.89E−04
F7	4.17E+07 ± 8.94E+06	5.36E+05 ± 1.05E+04	4.27E+06 ± 1.07E+05	1.19E+05 ± 8.91E+04	3.21E+04 ± 6.33E+03	6.63E+03 ± 4.56E+02
F8	4.29E+06 ± 5.39E+05	**1.12E+05 ± 3.16E+04**	4.32E+06 ± 9.65E+05	7.32E+05 ± 4.28E+04	9.07E+06 ± 8.16E+05	4.58E+05 ± 1.05E+04
F9	3.85E+09 ± 1.07E+08	3.47E+09 ± 4.97E+08	1.29E+09 ± 7.22E+08	4.46E+07 ± 3.36E+06	4.40E+08 ± 5.70E+07	7.16E+06 ± 7.69E+05
F10	4.77E+09 ± 3.14E+08	6.31E+08 ± 5.51E+07	7.66E+09 ± 5.14E+08	2.11E+09 ± 8.16E+08	2.63E+08 ± 3.28E+07	2.47E+07 ± 1.58E+06
F11	2.54E+09 ± 5.66E+08	7.02E+08 ± 2.23E+07	2.45E+09 ± 7.98E+08	7.56E+08 ± 4.70E+07	1.35E+07 ± 1.47E+06	5.38E+06 ± 2.34E+05
F12	1.36E+08 ± 2.47E+07	1.19E+09 ± 1.41E+08	1.98E+07 ± 4.21E+06	4.18E+06 ± 1.12E+05	8.44E+06 ± 7.28E+05	6.34E+06 ± 4.26E+05
F13	7.89E+00 ± 1.16E−01	7.32E+01 ± 5.90E+00	6.32E+00 ± 7.84E−01	9.97E+01 ± 6.68E+00	**1.25E−01 ± 2.30E−02**	4.48E+00 ± 3.90E+00
F14	6.33E+07 ± 3.58E+06	9.58E+06 ± 6.61E+05	4.14E+06 ± 9.65E+05	4.46E+06 ± 8.85E+05	3.47E+04 ± 4.28E+03	1.12E+04 ± 5.73E+03
F15	1.45E+06 ± 2.74E+05	1.88E+05 ± 3.47E+04	5.60E+05 ± 7.22E+04	7.48E+06 ± 4.32E+05	**2.19E+03 ± 1.27E+02**	7.75E+03 ± 4.16E+02
F16	3.69E+09 ± 1.08E+08	3.69E+08 ± 5.16E+07	6.37E+09 ± 5.36E+08	2.23E+07 ± 9.88E+07	7.65E+07 ± 3.21E+06	**9.33E+05 ± 5.36E+04**
F17	7.58E+02 ± 2.11E+01	5.17E+02 ± 4.35E+01	6.31E+02 ± 1.20E+01	4.88E+02 ± 4.73E+01	**3.21E+02 ± 7.85E+01**	4.21E+03 ± 7.48E+02
F18	5.33E+08 ± 9.36E+07	7.74E+07 ± 2.21E+06	5.22E+08 ± 8.45E+07	2.04E+07 ± 2.56E+06	4.58E+05 ± 1.24E+04	3.27E+04 ± 2.50E+03
F19	1.25E+04 ± 7.85E+03	9.85E+03 ± 7.75E+02	7.19E+04 ± 3.19E+03	1.39E+04 ± 1.28E+03	7.39E+04 ± 7.36E+03	1.16E+04 ± 1.16E+03
F20	2.64E+03 ± 2.45E+02	1.32E+03 ± 4.16E+02	2.34E+03 ± 2.40E+02	5.41E+04 ± 6.30E+03	2.14E+02 ± 3.49E+01	**8.42E+01 ± 3.65E+00**
F21	3.61E+03 ± 3.16E+02	4.51E+02 ± 1.12E+01	1.83E+03 ± 3.42E+02	7.66E+03 ± 1.15E+02	4.80E+02 ± 7.21E+01	**3.08E+01 ± 7.26E+00**
F22	8.96E+02 ± 7.48E+01	8.66E+01 ± 5.30E+00	7.69E+01 ± 5.96E+00	6.32E+02 ± 7.96E+01	**1.38E+01 ± 1.39E+00**	3.61E+01 ± 1.33E+00
F23	1.02E+08 ± 3.03E+07	4.32E+08 ± 7.85E+07	5.45E+08 ± 6.78E+07	5.49E+07 ± 7.69E+06	1.07E+09 ± 2.33E+08	4.97E+07 ± 7.58E+06
F24	4.36E+06 ± 2.75E+05	**1.08E+05 ± 1.19E+04**	9.74E+06 ± 9.65E+05	7.80E+06 ± 3.58E+05	4.94E+05 ± 1.00E+04	2.23E+06 ± 1.16E+05
F25	2.11E+08 ± 1.39E+07	7.30E+07 ± 6.33E+06	4.33E+08 ± 1.47E+07	6.33E+07 ± 1.89E+06	2.34E+05 ± 2.16E+04	1.18E+05 ± 4.52E+04
F26	3.75E+02 ± 7.45E+01	8.44E+02 ± 2.17E+01	8.14E+02 ± 3.58E+01	8.55E+02 ± 2.54E+01	7.21E+02 ± 4.25E+01	3.90E+03 ± 1.56E+02
F27	4.29E+08 ± 6.35E+07	9.84E+08 ± 6.60E+07	6.39E+08 ± 4.80E+07	4.31E+07 ± 3.74E+05	1.05E+07 ± 3.17E+06	**5.78E+06 ± 8.82E+05**
F28	1.71E+09 ± 4.17E+08	1.39E+08 ± 4.48E+07	7.41E+09 ± 6.31E+08	9.65E+08 ± 2.25E+07	1.06E+07 ± 3.75E+06	6.38E+06 ± 3.14E+05
CPU time	134.61	135.82	189.12	195.31	115.30	120.70

Here [a ± b] indicates the mean and standard deviation of 25 independent simulations. The best result for each function is shown in bold font. Average CPU times (minutes) are given in the last row of the table

Table 5 Comparison of the best error values of the 2013 CEC continuous benchmark functions with $D = 50$ for ABC, GSA, PSO, and their advanced versions

Function	ABC	MABC	GSA	CGSA2	PSO	LPSO	CPSO
F1	5.96E−17 ±9.20E−18	7.90E−18 ± 9.94E−19	4.38E−24 ± 6.19E−25	**2.81E−24 ± 7.89E−25**	3.35E−19 ± 3.44E−20	2.15E−15 ± 3.78E−16	5.53E−16 ± 1.274E−17
F2	3.74E+02 ±6.37E+01	**6.42E+01 ± 2.32E+00**	1.86E+02 ± 3.21E+01	1.15E+02 ± 2.25E+01	7.36E+03 ± 1.39E+02	4.12E+03 ± 2.44E+02	8.04E+04 ± 4.32E+02
F3	1.31E+01 ±1.56E+00	1.39E+01 ± 4.44E+00	8.01E+01 ± 7.25E+00	3.46E+03 ± 3.47E+02	1.43E+01 ± 2.38E+00	2.04E+01 ± 5.42E+00	5.52E+01 ± 7.32E+00
F4	4.40E−04 ±3.52E−05	2.30E−03 ± 8.63E−02	9.49E−04 ± 3.16E−05	**2.98E−04 ± 1.16E−05**	6.55E+00 ± 1.66E−01	4.36E−02 ± 1.81E−03	5.36E−04 ± 2.47E−05
F5	**1.83E−17 ± 4.19E−18**	5.44E−16 ± 1.09E−15	9.01E−16 ± 3.27E−17	4.74E−14 ± 1.05E−13	4.26E−08 ± 5.44E−09	4.24E−12 ± 5.23E−13	5.64E−15 ± 1.18E−16
F6	3.71E−03 ±2.27E−04	**1.96E−03 ± 3.45E−04**	3.47E−01 ± 5.36E−02	2.58E−01 ± 9.97E−02	5.90E+02 ± 3.28E+01	1.21E−01 ± 7.20E−02	7.63E−02 ± 4.15E−03
F7	2.06E+03 ±1.65E+04	**6.37E+02 ± 7.76E+01**	8.66E+04 ± 3.15E+03	1.72E+03 ± 3.25E+02	8.34E+07 ± 4.16E+06	9.30E+05 ± 5.77E+04	7.65E+05 ± 1.96E+04
F8	5.99E+06 ±3.32E+05	1.41E+06 ± 2.82E+05	8.64E+05 ± 7.85E+04	9.34E+05 ± 1.16E+04	4.51E+06 ± 5.89E+05	5.36E+07 ± 2.44E+06	6.35E+06 ± 2.28E+05
F9	1.34E+07 ±4.42E+06	**5.26E+06 ± 9.93E+05**	8.79E+08 ±3.27E+07	9.90E+08 ± 8.34E+07	7.82E+10 ± 7.33E+09	7.26E+09 ± 5.13E+08	2.28E+08 ± 7.65E+07
F10	2.47E+07 ±1.65E+06	**7.84E+06 ± 1.18E+05**	2.58E+08 ± 4.40E+07	3.68E+07 ± 2.48E+06	9.16E+10 ± 2.16E+09	7.31E+08 ± 5.66E+07	4.43E+08 ± 7.70E+07
F11	1.62E+08 ±3.78E+07	1.06E+07 ± 3.52E+06	9.66E+07 ± 5.41E+06	**1.14E+06 ± 7.29E+05**	8.01E+09 ± 4.58E+08	7.75E+06 ± 2.31E+05	4.32E+07 ± 2.36E+06
F12	3.11E+05 ±5.54E+04	**9.87E+04 ± 7.72E+03**	1.85E+06 ±4.36E+05	3.75E+05 ± 1.05E+04	6.63E+06 ± 3.29E+05	7.59E+06 ±2.16E+05	4.21E+06 ± 2.47E+05
F13	4.16E+00 ±6.32E−01	9.16E+01 ± 3.50E+00	9.50E+00 ± 3.27E−01	9.85E+00 ± 7.62E+00	1.19E+01 ± 1.76E+00	2.33E+01 ± 1.02E+00	7.75E−01 ± 2.26E−00
F14	5.32E+04 ±7.84E+03	**2.23E+03 ± 7.42E+02**	8.41E+05 ± 5.12E+04	4.23E+04 ± 1.18E+03	8.21E+09 ± 4.44E+08	7.66E+06 ± 2.34E+05	2.66E+05 ± 1.35E+04
F15	3.74E+04 ±2.14E+03	4.85E+03 ± 3.84E+02	9.87E+03 ± 3.16E+02	7.84E+03 ± 3.45E+02	7.76E+05 ± 5.34E+04	1.23E+05 ± 6.33E+04	8.65E+06 ± 3.32E+05
F16	1.85E+06 ±5.50E+05	3.90E+06 ± 3.95E+05	7.65E+07 ±5.39E+06	1.12E+07 ± 2.47E+06	2.17E+08 ± 1.65E+07	2.17E+09 ± 1.65E+08	5.04E+07 ± 3.28E+06
F17	5.32E+04 ±3.96E+03	7.75E+04 ± 1.07E+03	6.10E+02 ± 3.89E+01	7.36E+02 ± 3.40E+01	3.43E+02 ± 2.19E+01	9.65E+02 ± 3.28E+01	7.65E+03 ± 4.11E+02
F18	6.11E+04 ±2.28E+03	**2.60E+03 ± 6.62E+02**	2.04E+05 ±1.58E+04	5.44E+04 ± 5.54E+03	8.92E+09 ± 8.75E+08	5.59E+07 ± 7.74E+06	1.76E+06 ± 3.14E+05
F19	7.19E+04 ±1.65E+04	1.34E+04 ± 7.41E+03	1.02E+04 ± 7.66E+03	**6.90E+03 ± 2.61E+02**	9.08E+05 ± 2.15E+05	1.05E+05 ± 6.61E+04	5.65E+04 ± 4.41E+03
F20	1.50E+02 ±7.85E+01	5.80E+02 ± 6.95E+01	8.42E+02 ± 5.29E+01	7.86E+02 ± 3.45E+01	7.71E+03 ± 4.43E+02	5.32E+03 ± 4.17E+02	2.39E+03 ± 6.40E+02
F21	6.33E+03 ±4.26E+02	3.26E+03 ± 1.16E+02	9.45E+03 ±3.76E+02	4.30E+03 ± 7.76E+02	2.59E+04 ± 3.29E+03	1.86E+04 ±5.24E+03	6.32E+03 ± 1.76E+02
F22	1.36E+02 ±8.91E+01	7.13E+01 ± 2.25E+00	8.33E+01 ± 3.72E+00	7.85E+01 ± 1.46E+00	2.34E+02 ± 1.38E+02	7.26E+01 ± 9.01E+00	4.99E+01 ± 2.40E+00
F23	**2.47E+07 ± 6.33E+06**	9.77E+07 ± 7.74E+06	9.85E+09 ± 2.28E+08	3.20E+08 ± 3.27E+07	8.93E+09 ± 2.16E+08	5.33E+07 ± 7.18E+06	2.66E+09 ± 5.14E+08
F24	3.84E+05 ±5.47E+04	6.34E+07 ± 3.60E+06	8.40E+05 ± 5.37E+04	1.12E+05 ± 1.19E+04	9.32E+06 ± 4.43E+05	1.01E+07 ± 3.47E+06	2.61E+07 ± 1.96E+06
F25	1.02E+05 ±2.08E+04	**1.08E+04 ± 8.87E+03**	1.09E+05 ± 3.38E+04	2.57E+04 ± 2.85E+03	6.89E+05 ± 5.23E+04	3.24E+06 ± 5.17E+05	8.82E+07 ± 7.43E+06
F26	3.44E+02 ±1.19E+01	3.65E+02 ± 1.15E+01	**1.01E+02 ± 716E+01**	4.31E+02 ± 4.46E+01	1.90E+03 ± 1.77E+02	6.47E+03 ± 1.56E+02	8.77E+04 ± 1.48E+03
F27	9.07E+06 ±3.50E+05	4.43E+07 ± 3.62E+06	7.74E+07 ± 6.08E+06	8.16E+06 ± 3.71E+05	5.76E+07 ± 4.10E+06	3.69E+08 ± 2.38E+07	1.26E+08 ± 2.35E+07
F28	5.16E+08 ±7.81E+07	**1.12E+06 ± 7.71E+05**	7.31E+08 ± 5.46E+07	4.45E+07 ± 2.05E+05	3.11E+09 ± 5.43E+08	1.16E+07 ± 3.22E+06	6.48E+08 ± 7.71E+07
CPU time	142.07	154.95	165.24	187.63	155.04	172.02	185.13

Here [a ± b] indicates the mean and standard deviation of 25 independent simulations. The best result for each function is shown in bold font. Average CPU times (minutes) are given in the last row of the table

Table 6 Friedman ranks of all SI algorithms for the 2013 CEC benchmark functions

Function	SFLA	MSFLA	GSO	SEGSO	FA	VSSFA	ABC	MABC	GSA	CGSA2	PSO	LPSO	CPSO
F1	11	9	13	12	4	3	7	6	2	1	5	10	8
F2	8	5	13	12	7	2	6	1	4	3	10	9	11
F3	7	10	8	12	2	1	3	4	11	13	5	6	9
F4	13	11	12	10	4	3	7	2	6	1	9	8	5
F5	11	13	12	10	6	2	1	3	4	7	9	8	5
F6	12	13	9	10	4	3	2	1	8	7	11	6	5
F7	12	8	11	7	5	4	3	1	6	2	9	10	13
F8	7	1	8	3	12	2	10	6	4	5	11	13	9
F9	11	10	9	4	6	2	3	1	7	8	13	12	5
F10	11	8	12	10	6	2.5	2.5	1	5	4	13	9	7
F11	12	9	11	10	5	2	8	4	7	1	13	3	6
F12	12	13	11	5	10	7	2	1	4	3	8	9	6
F13	6	11	5	13	1	4	3	12	7	8	9	10	2
F14	12	11	8	9	3	2	5	1	7	4	13	10	6
F15	11	8	9	12	1	3	6	2	5	4	10	7	13
F16	12	10	13	5	7.5	1	2	3	7.5	4	9	11	6
F17	8	4	6	3	1	10	12	13	5	7	2	9	11
F18	2	10	11	8	6	12	4	1	5	3	13	9	7
F19	5	2	9.5	7	11	4	9.5	6	3	1	13	12	8
F20	10	7	8	13	3	1	2	4	6	5	12	9	11
F21	6	2	4	10	3	1	9	5	11	7	13	12	8
F22	13	9	6	12	1	2	4	10	8	7	11	5	3
F23	6	8	9	4	10	2	1	5	13	7	12	3	11
F24	7	1	10	8	4	6	3	13	5	2	9	11	12
F25	12	5	13	9	6	10	3	1	4	2	7	8	11
F26	4	8	7	9	6	11	2	3	1	5	10	12	13
F27	11	13	12	5	4	1	3	6	8	2	7	10	9
F28	11	6	13	10	3	2	7	1	9	5	12	4	8
Statistic		130.45					p value		0.00012				

"Statistic" and "p value" in the last row indicate Friedman statistic and corresponding p value, respectively

Table 7 Holm multiple comparison test results of PSO and other SI algorithms, which show the average rank and the p values

Algorithm	Rank	p value	Algorithm	Rank	p value
SFLA	9.67	0.08121	ABC	4.86	0.00009
MSFLA	7.89	0.00517	MABC	3.62	0.00002
GSO	9.75	0.08902	GSA	6.12	0.00094
SEGSO	8.62	0.02341	CGSA2	4.57	0.00008
FA	5.01	0.00017	LPSO	8.75	0.04536
VSSFA	4.14	0.00005	CPSO	8.18	0.00914

PSO is the control algorithm, and its average rank is 9.93 based on the Friedman test (not shown in the table)

of the algorithm, and this effect is problem-dependent. Second, if we use more other advanced versions of SFLA, GSO, FA, ABC, and GSA, we might obtain different results. The purpose of the comparisons here is not to tune the parameters of the algorithms to obtain the best performance, but rather to quantify performance differences between typically implemented algorithm versions, and to show that the algorithmic differences between SI algorithms can result in significantly different performance.

3.2 Combinatorial knapsack problems

In this section the SI algorithms are applied to the knapsack problems, which is an important and representative real-world combinatorial problem (Abulkalamazad et al. 2014; Bhattacharjee and Sarmah 2014; Freville 2004). Many combinatorial problems can be reduced to a knapsack problem.

The 0/1 knapsack problem can be simply described as follows. Consider a set of n items, where the ith item has weight w_i and profit p_i. The problem is to select a subset of the n items to maximize overall profit without exceeding

the weight constraint b. The problem can be mathematically modeled as follows:

$$\begin{aligned} &\text{Maximize} \quad \sum_{i=1}^{n} w_i x_i \\ &\text{Subject to} \quad \sum_{i=1}^{n} p_i x_i \leq b, \\ &\text{where } x_i \in \{0,\ 1\} \quad \forall \quad i \in \{1, 2, \ldots, n\} \end{aligned} \tag{9}$$

x_i takes the value of 1 or 0, which represents the selection or rejection of the ith item. Ten benchmark knapsack problems are studied here, as summarized in Table 8. The parameters used in the SI algorithms in this section are the same as those in Sect. 3.1.

Table 9 shows comparisons of the best performance of each SI algorithm and their advanced versions after 1000 generations, averaged over 25 simulations. The results show that all SI algorithms and their advanced versions perform the same for f_3, f_4, and f_9; SFLA and its advanced version obtain the best performance for five of the problems; GSA and its advanced version obtain the best performance for four and five of the problems, respectively; and FA, ABC and their advanced versions obtain the best performance for two of the problems. This indicates that SFLA and GSA are significantly better than the other SI algorithms for the combinatorial problems that we study. These results are also consistent with the observation in Table 1, which indicates that SFLA is the most appropriate for combinato-

Table 8 The dimension and parameters of the ten benchmark knapsack problems

	Dim.	Parameters (w, p, b)
f_1	10	w = {95, 4, 60, 32, 23, 72, 80, 62, 65, 46}; p = {55, 10, 47, 5, 4, 50, 8, 61, 85, 87}; b = 269
f_2	20	w = {92, 4, 43, 83, 84, 68, 92, 82, 6, 44, 32, 18, 56, 83, 25, 96, 70, 48, 14, 58};
		p = {44, 46, 90, 72, 91, 40, 75, 35, 8, 54, 78, 40, 77, 15, 61, 17, 75, 29, 75, 63}; b = 878
f_3	4	w = {9, 11, 13, 15}; p = {6, 5, 9, 7}; b = 20
f_4	4	w = {6, 10, 12, 13}; p = {2, 4, 6, 7}; b = 11
f_5	15	w = {56.358531, 80.87405, 47.987304, 89.59624, 74.660482, 85.894345, 51.353496, 1.498459, 36.445204, 16.589862, 44.569231, 0.466933, 37.788018, 57.118442, 60.716575};
		p = {0.125126, 19.330424, 58.500931, 35.029145, 82.284005, 17.41081, 71.050142, 30.399487, 9.140294, 14.731258, 98.852504, 11.908322, 0.89114, 53.166295, 60.176397};b = 375
f_6	10	w = {30, 25, 20, 18, 17, 11, 5, 2, 1, 1}; p = {20, 18, 17, 15, 15, 10, 3, 1, 1}; b = 60
f_7	7	w = {70, 20, 39, 37, 7, 5, 10}; p = {31, 10, 20, 19, 4, 3, 6}; b = 50
f_8	23	w = {983, 982, 981, 980, 979, 978, 488, 976, 972, 486, 486, 972, 972, 485, 485, 969, 966, 483, 964, 963, 961, 958, 959};
		p = {81, 980, 979, 978, 977, 976, 487, 974, 970, 485, 485, 970, 970, 484, 484, 976, 974, 482, 962, 961, 959, 958, 857}; b = 10000
f_9	5	w = {33, 24, 36, 37, 12}; p = {15, 20, 17, 8, 31}; b = 80
f_{10}	20	w = {84, 83, 43, 4, 44, 6, 82, 92, 25, 83, 56, 18, 58, 14, 48, 70, 96, 32, 68, 92};
		p = {91, 72, 90, 46, 55, 8, 35, 75, 61, 15, 77, 40, 63, 75, 29, 75, 17, 78, 40, 44}; b = 879

Table 9 Performance comparison for six SI algorithms and their advanced versions on benchmark knapsack problems

	SFLA	MSFLA	GSO	SEGSO	FA	VSSFA	ABC	MABC	GSA	CGSA2	PSO	LPSO	CPSO
f_1	**295**	**295**	265	276	**295**	**295**	**295**	**295**	**295**	**295**	268	288	280
f_2	**950**	**950**	865	904	932	942	937	940	954	**950**	836	910	894
f_3	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**	**35**
f_4	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**	**23**
f_5	466	472	395	400	444	450	435	445	**481**	**481**	384	423	405
f_6	**52**	**52**	49	50	**52**	**52**	**52**	**52**	**52**	**52**	49	51	50
f_7	107	**108**	105	107	106	106	106	107	107	107	105	105	105
f_8	**9760**	**9760**	9724	9750	9748	9750	9750	9755	9759	9759	9730	9752	9738
f_9	**130**	**130**	**130**	**130**	**130**	**130**	**130**	**130**	**130**	**130**	**130**	**130**	**130**
f_{10}	**1010**	**1010**	870	890	982	990	1002	1008	**1010**	**1010**	896	975	937

The numbers show the best performance, averaged over 25 simulations. The best performance in each row is shown in bold font

rial optimization problems, and that GSA can obtain good performance for most types of optimization problems.

4 Conclusions

The similarities and differences of several popular new swarm intelligence algorithms, including SFLA, GSO, FA, ABC, and GSA, have been discussed in detail. Our discussion and comparison has been based on algorithmic motivations and implementation details. These algorithms have many similarities with PSO due to their similar biological motivations. We found that SFLA, GSO, FA, ABC and GSA are equivalent to PSO under certain conditions. In addition, we compared SFLA, GSO, FA, ABC, and GSA, with the basic PSO and with their improved variations, on a set of continuous benchmark functions and combinatorial knapsack problems. Simulation results show that although the algorithms are identical under certain conditions, their performance levels are quite different under the tested conditions, because each algorithm retains its own distinctions when implemented in its standard form. Given the overlap between algorithms, it is often difficult to know when one SI algorithm ends and another begins, when a new SI algorithm deserves to belong to its own class, or when it should be classified as a variation of an existing SI algorithm. We conclude that it can be helpful to maintain the distinctions between various SI algorithms, because the plethora of algorithms provides a diverse choice of optimization methods, research opportunities, and application opportunities.

SI researchers and practitioners typically want to know which algorithm performs best. However, the no-free-lunch theorem and the simulation results in this paper show that this is a poorly framed question. Relative performance depends on the particular problem, and is greatly affected by algorithmic variations and tuning parameters. One of the challenges for the SI research community is to find a balance between encouraging new research directions while still maintaining high standards for the introduction and development of purportedly new SI algorithms. However, the empirical results in this paper show that the advanced version of ABC performs best on continuous benchmark functions, and the advanced versions of SFLA and GSA perform best on the combinatorial knapsack problems. These results contrast with previous publications. For instance, (Elbeltagi et al. 2005) indicates that PSO performs better than GA, MA, ACO, and SFL; (Davarynejad et al. 2014) indicates that GSA performs better than PSO; (Parpinelli et al. 2012) indicates that PSO performs better than BFO and ABC; and (Civicioglu and Besdok 2013) indicates that cuckoo search performs better than PSO and ABC. These divergent results simply confirm our hypothesis that performance depends strongly on algorithmic variations and problem selection.

For future work there are several important directions. Many other SI algorithms exist, such as GWO, BA, FWA, and variations, which could provide better performance for certain types of problems than the algorithms studied in this paper. It would be interesting to analyze their similarities and differences also. The second suggested direction for future research is to study theoretical similarities and differences between algorithms based on mathematical models such as Markov chains (Nix and Vose 1992; Suzuki 1995), dynamic systems (Simon 2011), and statistical mechanics (Ma et al. (2015)). This effort would provide more definite mathematical conclusions than simulation results. The third suggested direction for future work is to develop improved versions of SI algorithms by using natural principles.

Acknowledgements This material is based upon work supported by the National Science Foundation under Grant No. 1344954, the National Natural Science Foundation of China under Grant Nos. 61305078, 61533010, 61179041.

Compliance with ethical standards

Conflict of interest The authors declare that there is no conflict of interests regarding the publication of this paper.

References

Abulkalamazad M, Rocha A, Fernandes E (2014) Improved binary artificial fish swarm algorithm for the 0–1 multidimensional knapsack problems. Swarm Evolut Comput 14:66–75

Bahriye A, Dervis K (2012) A modified artificial bee colony algorithm for real-parameter optimization. Inf Sci 192:120–142

Bhattacharjee K, Sarmah SP (2014) Shuffled frog leaping algorithm and its application to 0/1 knapsack problem. Appl Soft Comput 19:252–263

Chen D, Wang J, Zou F, Hou W, Zhao C (2012) An improved group search optimizer with operation of quantum-behaved swarm and its application. Appl Soft Comput 12:712–725

Civicioglu P, Besdok E (2013) A conceptual comparison of the Cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms. Artif Intell Rev 39:315–345

Clerc M, Kennedy J (2002) The particle swarm-explosion, stability and convergence in a multidimensional complex space. IEEE Trans Evolut Comput 6(3):58–73

Cobos C, Muñoz-Collazos H, Urbano-Muñoz R, Mendoza M, León E, Herrera-Viedma E (2014) Clustering of web search results based on the cuckoo search algorithm and balanced Bayesian information criterion. Inf Sci 281:248–264

Davarynejad M, Berg J, Rezaei J (2014) Evaluating center-seeking and initialization bias: the case of particle swarm and gravitational search algorithms. Inf Sci 278:802–821

Derrac J, Garcia S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evolut Comput 1:3–18

Dervis K, Bahriye B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Glob Optim 39:459–471

Dervis K, Beyza G, Celal O, Nurhan K (2014) A comprehensive survey: artificial bee colony (ABC) algorithm and applications. Artif Intell Rev 42:21–57

Dorigo M, Gambardella LM (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans Evolut Comput 1(3):53–66
Dowlatshahi MB, Nezamabadi-pour H, Mashinchi M (2014) A discrete gravitational search algorithm for solving combinatorial optimization problems. Inf Sci 258:94–107
Elbeltagi E, Hegazy T, Grierson D (2005) Comparison among five evolutionary-based optimization algorithms. Adv Eng Inform 19:43–53
Emad E, Tarek H, Donald G (2007) A modifed shuffled frog-leaping optimization algorithm: applications to project management. Struct Infrastruct Eng 3(1):53–60
Eusuff M, Lansey KE (2003) Optimization of water distribution network design using the shuffled frog leaping algorithm. J Water Res Pl-ASCE 129:210–225
Fister I, Jr Fister, Yang X, Brest J (2013) A comprehensive review of firefly algorithms. Swarm Evolut Comput 13:34–46
Freville A (2004) The multidimensional 0–1 knapsack problem: an overview. Eur J Oper Res 155:1–20
Gao S, Vairappan C, Wang Y, Cao Q, Tang Z (2014) Gravitational search algorithm combined with chaos for unconstrained numerical optimization. Appl Math Comput 231:48–62
Hasançebi O, Carbas S (2014) Bat inspired algorithm for discrete size optimization of steel frames. Adv Eng Softw 67:173–185
He S, Wu Q, Saunders J (2006) A novel group search optimizer inspired by animal behavioral ecology. In: Proceedings of the IEEE international conference on evolutionary computation, pp 1272–1278
Jiang S, Wang Y, Ji Z (2014) Convergence analysis and performance of an improved gravitational search algorithm. Appl Soft Comput 24:363–384
Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Technical report, Computer Engineering Department, Erciyes University
Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Glob Optim 39:459–471
Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of the IEEE international conference on neural networks, pp 1942–1948
Koza JR (1992) Genetic programming: on the programming of computers by means of natural selection. MIT Press, Cambridge
Krishnanand KN, Ghose D (2009) Glowworm swarm optimization for simultaneous capture of multiple local optima of multimodal functions. Swarm Intell 3:87–124
Liao T, Stuetzle T (2013) Benchmark results for a simple hybrid algorithm on the CEC 2013 benchmark set for real parameter optimization. In: Proceedings of IEEE congress on evolutionary computation, pp 1938–1944
Ma H, Simon D, Fei M (2015) On the statistical mechanics approximation of biogeography-based optimization. Evolut Comput. doi:10.1162/EVCO_a_00160
Mirjalili S, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
Nix A, Vose M (1992) Modeling genetic algorithms with Markov chains. Ann Math Intell 5:79–88
Parpinelli R, Lopes H (2011) New inspirations in swarm intelligence: a survey. Int J Bio-Inspired Comput 3:1–16
Parpinelli R, Teodoro F, Lopes H (2012) A comparison of swarm intelligence algorithms for structural engineering optimization. Int J Numer Methods Eng 19:666–684
Rahimi-Vahed A, Mirzaei A (2008) Solving a bi-criteria permutation flow-shop problem using shuffled frog-leaping algorithm. Soft Comput 12:435–452
Rao RV, Savsani VJ, Vakharia DP (2011) Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems. Comput-Aided Des 43:303–315
Rashedi E, Nezamabadi-pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179:2232–2248
Rashedi E, Nezamabadi-pour H, Saryazdi S (2011) Filter modeling using gravitational search algorithm. Eng Appl Artif Intell 24:117–122
Reeves C, Rowe J (2003) Genetic algorithms: principles and perspectives. Kluwer Academic Publishers, Boston
Sarkheyli A, Zain AM, Sharif S (2015) The role of basic, modified and hybrid shuffled frog leaping algorithm on optimization problems: a review. Soft Comput 19:2011–2038
Schwefel HP (1995) Evolution and optimum seeking. Wiley Press, New Jersey
Simon D (2011) A dynamic system model of biogeography-based optimization. Appl Soft Comput 11:5652–5661
Simon D (2013) Evolutionary optimization algorithms. Wiley, New Jersey
Shen H, Zhu Y, Niu B, Wu Q (2009) An improved group search optimizer for mechanical design optimization problems. Prog Nat Sci 19:91–97
Shi Y, Eberhart RC (1998) Parameter selection in particle swarm optimization. Lect Notes Comput Sci 1447:591–600
Suzuki J (1995) A Markov chain analysis on simple genetic algorithms. IEEE Trans Syst Man Cybern Part B 25:655–659
Tan Y, Zhu Y (2010) Fireworks algorithm for optimization. Lect Notes Comput Sci 6145:355–364
Wang L, Fang C (2011) An effective shuffled frog-leaping algorithm for multi-mode resource-constrained project scheduling problem. Inf Sci 181:4804–4822
Wang L, Zhong X, Liu M (2012) A novel group search optimizer for multi-objective optimization. Expert Syst Appl 39:2939–2946
Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evolut Comput 1:67–82
Yang XS (2009) Firefly algorithms for multimodal optimization. Lect Notes Comput Sci 5792:169–178
Yang X (2011) Review of meta-heuristics and generalized evolutionary walk algorithm. Int J Bio-Inspired Comput 3:77–84
Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evolut Comput 3(1):82–102
Yu S, Zhu S, Ma Y, Mao D (2015) A variable step size firefly algorithm for numerical optimization. Appl Math Comput 263:214–220
Zang H, Zhang S, Hapeshi K (2010) A review of nature-inspired algorithms. J Bionic Eng 7(Supplement):S232–S237
Zare K, Haque M, Davoodi E (2012) Solving non-convex economic dispatch problem with valve point effects using modified group search optimizer method. Electr Power Syst Res 84:83–89
Zheng X, Lu D, Chen Z (2014) A self-adaptive group search optimizer with elitist strategy. In: Proceedings of 2014 IEEE congress on evolutionary computation, pp 2033–2039