Expert Systems with Applications 41 (2014) 6555–6569

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

A modified Intelligent Water Drops algorithm and its application to optimization problems

Basem O. Alijla [a,b,*], Li-Pei Wong [a], Chee Peng Lim [c], Ahamad Tajudin Khader [a], Mohammed Azmi Al-Betar [a,d]

[a] School of Computer Sciences, Universiti Sains Malaysia, Malaysia
[b] Faculty of Information Technology, Islamic University of Gaza, Palestine
[c] Centre for Intelligent Systems Research, Deakin University, Australia
[d] Department of Information Technology, Al-Balqa Applied University, Jordan

ARTICLE INFO

Article history:
Available online 20 May 2014

Keywords:
Intelligent Water Drops (IWD)
Swarm-based optimization
Ranking-based selection methods
Feature selection (FS)
Rough set (RS)
Multiple knapsack problem (MKP)
Travelling salesman problem (TSP)

ABSTRACT

The Intelligent Water Drop (IWD) algorithm is a recent stochastic swarm-based method that is useful for solving combinatorial and function optimization problems. In this paper, we investigate the effectiveness of the selection method in the solution construction phase of the IWD algorithm. Instead of the fitness proportionate selection method in the original IWD algorithm, two ranking-based selection methods, namely linear ranking and exponential ranking, are proposed. Both ranking-based selection methods aim to solve the identified limitations of the fitness proportionate selection method as well as to enable the IWD algorithm to escape from local optima and ensure its search diversity. To evaluate the usefulness of the proposed ranking-based selection methods, a series of experiments pertaining to three combinatorial optimization problems, i.e., rough set feature subset selection, multiple knapsack and travelling salesman problems, is conducted. The results demonstrate that the exponential ranking selection method is able to preserve the search diversity, therefore improving the performance of the IWD algorithm.

1. Introduction

Optimization is concerned with finding the best solution for a given problem in the search space (Weise & et al., 2009). Optimization problems are confronted in numerous domains, for instance optimizing the outcome of mathematical functions within a range of variables (Vesterstrom & Thomsen, 2004); optimizing computing and engineering system performances (Machado et al., 2001; Marler & Arora, 2004); optimizing cost, profit, and product quality in the manufacturing industry (Yildiz, 2009). All these optimization problems entail the same fundamental principle, i.e., choosing an alternative that maximizes or minimizes the objective function subject to certain constraints, i.e. $\min/\max\{f(x) \in \mathbf{X}\}$, where $f(x)$ is the objective function, $\boldsymbol{x} = \{\boldsymbol{x_i} | i = 1, \ldots, N\}$ is a set of alternatives, i.e. decision variables, $\mathbf{X} = \{\mathbf{X_i} | i = 1, \ldots, N\}$ is the possible set of values that the decision variable can take, and N is the number of decision variables.

In general, optimization problems can be classified into two types: function optimization and combinatorial optimization (Weise et al., 2009). The former, also known as continuous optimization, deals with selecting the appropriate values of variables, which are defined over the continuous range. The latter refers to discrete problems in which not only the value but also the ordering of variables is important, e.g. travelling salesman and scheduling problems.

Many methods have emerged to tackle optimization problems. They can be divided into two categories based on the produced solutions (Weise et al., 2009), namely: deterministic and non-deterministic methods. Deterministic (exact) methods provide the same solution in different runs. On the other hand, non-deterministic (stochastic) methods exhibit some randomness, and produce different solutions in different runs. Deterministic methods comprise a variety of approaches which include some local search methods such as linear programming, branch-and-bound, and best-first search. One limitation of these methods is the search process can be trapped in a local optimum. On the other hand, non-deterministic methods explore several regions of the search space at the same time, and have the ability to escape from the local optimum and reach a global maximum or minimum point, Therefore, non-deterministic methods are more appropriate to tackle NP-hard problems (i.e. problems that have no known solutions in polynomial time) (Lin, Tsai, & Yu, 2012).

* Corresponding author at: School of Computer Sciences, Universiti Sains Malaysia, Malaysia. Tel.: +60 142426175.
E-mail address: bofa10_com026@student.usm.my (B.O. Alijla).

http://dx.doi.org/10.1016/j.eswa.2014.05.010
0957-4174/

In the past two decades, a new branch of non-deterministic optimization methods known as nature-inspired models have emerged (Yang, 2008). Nature-inspired models are based on natural principles or behaviors, e.g. genetic algorithms are inspired by biological evolution of organisms such as inheritance, mutation, crossover, and selection (Srinivas & Patnaik, 1994). Recently, a new class of nature-inspired models, known as Swarm Intelligence (SI), which originates from the behaviors of different natural swarms, has been proposed. A number of SI-based optimization models, e.g. ant colony optimization inspired by the foraging behavior of real ants (Colorni et al., 1994; Dorigo & Di Caro, 1999), particle swarm optimization inspired by the social behavior of bird flocking or fish schooling (Shi, 2001), artificial bee colony inspired by the foraging behavior of honey bees in their colony (Karaboga, 2005; Wong, Low, & Chong, 2010a, 2010b), bacterial foraging inspired by the social foraging behavior of *Escherichia coli* (Passino, 2002), have been successfully used in solving different optimization problems. SI-based models are characterized by collaborative learning, i.e., a population of agents collaborates and cooperates among themselves within their environment to solve a problem. The interactions among agents enable the model to explore several regions of the search space simultaneously and reach the global optimum solution (Ahmed & Glasgow, 2012).

The Intelligent Water Drops (IWD) algorithm, which is a relatively recent SI-based model, was introduced by Shah-Hosseini (2007). As a constructive-based method, the IWD algorithm constructs an optimal solution through cooperation among a group of agents called water drops. The algorithm imitates the phenomena of a swarm of water drops flowing with soil along a river bed.

Procedurally, each water drop incrementally constructs a solution through a series of iterative transitions from one node to the next until a complete solution is obtained. Water drops communicate with each other through an attribute called soil, which is associated with the path between any two points. The soil value is used to determine the direction of movement from the current node to the next, whereby a path with a lower amount of soil is likely to be followed. The IWD algorithm introduced by Shah-Hosseini utilizes the fitness proportionate selection (FPS) method to identify the probability of transition from one node to the next, as shown in Eq. (1)

$$P(i) = \frac{f(i)}{\sum_{j=1}^{N} f(j)} \tag{1}$$

where $P(i)$ represents the probability of selecting node i, $f(\cdot)$ is the fitness function, and N is the number of candidate nodes. The fitness function of candidate node i is inversely proportional to the absolute soil value, i.e., $f(i) = \frac{1}{soil(current,i)}$.

The IWD algorithm has been successfully employed to solve a number of combinatorial and function optimization problems, such as function optimization, travelling salesman, multiple knapsack, n-queen puzzle problems (Shah-Hosseini, 2007, 2008, 2012a, 2012b). The success is attributed to its capability of memorizing the search experience through the soil and velocity variables that inherit better solutions from one generation to the next. However, there are three shortcomings in its FPS method, i.e. negative soil values, indistinguishable fitness, and domination by outstanding nodes (as explained and analyzed in the next). Motivated by the limitations of the FPS method, two ranking-based selection methods, i.e. linear and exponential ranking methods are proposed for the IWD algorithm in this paper. This forms the main contribution of this research. The modified IWD algorithm is compared with other algorithms using three benchmark optimization problems, i.e., rough set feature subset selection (RSFS) Jensen & Shen, 2003, multiple knapsack problem (MKP) Shah-Hosseini, 2008, and travelling salesman problem (TSP) Shah-Hosseini, 2007. The results are analyzed systematically and discussed comprehensively. The results demonstrate the usefulness of the proposed ranking-based selection methods in enhancing the performance of the original IWD algorithm with the FPS method. While the effectiveness of the proposed ranking-based selection methods can be observed, the shortcoming is that they are computationally more expensive than the current FPS method (as shown in the case studies in Section 5).

As stated earlier, the FPS method in the original IWD algorithm is susceptible to three limitations, i.e.:

A. Inability to accommodate negative soil values;
B. Inability to create a different selection pressure for fitter nodes when most of the nodes have a similar fitness value, i.e. indistinguishable fitness;
C. Inability to handle selection dominated by a node with an exceptionally high fitness value as compared with those from other nodes in a selection pool, i.e., domination by an outstanding node.

Fig. 1 shows an example of the scenario whereby a water drop resides in the current node, A, and intends to move to the next node, either B, C, D, E, or F. The number on the arc between two nodes represents the soil value. To select the next node, the selection probability of each node is computed according to Eq. (1). Based on the FPS method, node D has the highest probability to be selected.

Limitations B and C can be solved using a scaling function; but this makes the FPS method sensitive to the scaling function (Al-Betar et al., 2012, 2013). Even though a scaling function is applied, the indistinguishable probability can still be maintained. As an example, consider the soil values between node A and the others (as shown in Fig. 1) are scaled by adding a constant, $x = 1000$, to each value. The new selection probabilities are shown in Fig. 2. As can be seen, the selection probabilities of all nodes are indistinguishable, whereby the best and the worst nodes (i.e. D and F) have almost similar selection probabilities (Goldberg & Deb, 1991).

As for limitation C, consider the scenario shown in Fig. 3. The fittest node, D, dominates the others because its soil value has a large difference as compared with those from the others.

The aforementioned limitations are likely to occur in the IWD algorithm, where the soil can take negative values. A large difference between the soil values is possible due to the local update process, which is applied to every water drop movement, as well as the global update process, which is applied once. Since the soil update mechanism with the FPS method does not maintain diversity of the water drops, i.e. selecting the same node over and over again, a homogenous population is produced.

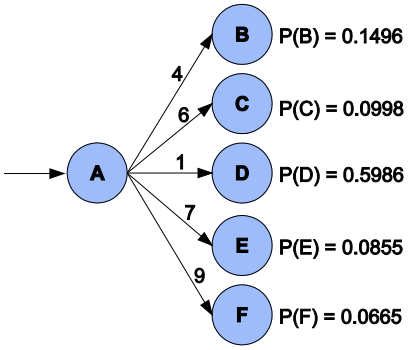

Fig. 1. An example showing the selection probabilities of five candidate nodes according to the FPS method.

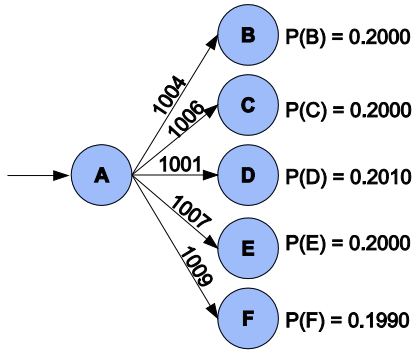

Fig. 2. Sensitivity of FPS method to the scaling function with a indistinguishable fitness value.

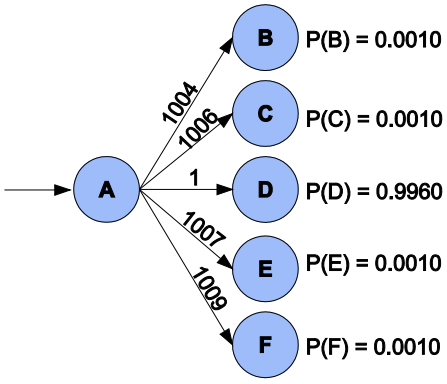

Fig. 3. Inability of the FPS method to handle the selection dominated by an outstanding node.

The rest of this paper is organized as follows. In Section 2, a number of related investigations on the IWD algorithm are reviewed. In Section 3, the fundamental principles of the IWD algorithm are explained. The proposed modifications to the original IWD algorithm are presented in Section 4. The experiments and the associated results of the proposed IWD algorithm are described in Section 5. Conclusions and suggestions for future research are given in Section 6.

2. Literature review

In 2007, the IWD algorithm was proposed, adding a new SI-based nature-inspired optimization to the literature (Shah-Hosseini, 2007). It is based on the observation of flowing water drops, which creates an optimal path between the upstream and downstream of a river. The IWD algorithm imitates some natural interactions between water drops and changes of the river environment. In the beginning, the IWD algorithm was employed to solve TSP (Shah-Hosseini, 2007). Over the past few years, it has been used successfully to solve some NP-hard optimization problems (Shah-Hosseini, 2009, 2012b). The success of the IWD algorithm stems from two salient properties: (i) its cooperative learning mechanism allows independent agents to exchange the search knowledge through the search environment and parameters; (ii) its robust learning mechanism enables the algorithm to memorize the search history. Most of the IWD applications are related to engineering and technological problems, e.g. multi-dimensional knapsack and n-queen puzzle (Shah-Hosseini, 2008, 2009), robot path planning (Duan, Liu, & Lei, 2008), vehicle routing (Akbarzadeh Totonchi, 2010), and economic load dispatch problems (Nagalakshmi & et al., 2011). In addition, the IWD algorithm is useful for tackling several machine learning and image process optimization problems, e.g. feature selection in multi-level thresholding of gray-level images (Hendrawan & Murase, 2011; Shah-Hosseini, 2012a).

While a lot of investigations have been conducted with respect to IWD applications, limited research efforts are focused on improving the theoretical aspects of the IWD algorithm. In Niu, Ong, and Nee (2012), an Enhanced IWD algorithm (EIWD) to solve the job-shop scheduling problems was proposed. Five schemes were introduced to increase diversity of the search space and enhance the performance of the original IWD algorithm, as follows:

(i) Diverse soil and velocity initialization, whereby the initial amount of soil for each edge and the initial velocity of every water drop are randomly selected;
(ii) Conditional probability computation, whereby the selection probability considers the processing time (operation) of the next node in order to increase diversity of the search space by introducing some degree of randomness
(iii) Bounded local soil update, in which the soil update is bounded by the lower and upper values that control the convergence rate;
(iv) Elite global soil update, in which all the soil values of all edges included in the elite group are updated;
(v) Combined local search, whereby a local search that integrates both breadth and depth search to improve the quality of the produced solution is introduced in the EIWD algorithm.

A recent paper reported the applicability of the IWD algorithm to continuous optimization problems (Shah-Hosseini, 2012b). The continuous real variables are encoded into a binary string. The allowable range of any decision variable is sampled into P samples. It represents the precision of the continuous real variable. To tailor the IWD algorithm for solving continuous optimization problems, every node in the construction graph is extended to consecutive P nodes so that a binary string with P bits for each variable is established. As such, an optimization problem with M variables leads to a construction graph with $M \times P$ nodes and $2 \times M \times P$ edges. Every water drop starts its journey to visit the next node by considering all the corresponding P nodes.

In this paper, we focus on the theoretical aspect of the selection method in the IWD algorithm. Specifically, two ranking-based selection methods are adopted in the solution construction phase of the IWD algorithm to overcome the limitations of the FPS method; therefore improving the IWD algorithm performance in tackling optimization problems. To the best of the authors' knowledge, this is the first attempt that investigates the effectiveness of the linear and exponential ranking selection methods in the solution construction phase of the IWD algorithm. The details are presented in the following sections.

3. The IWD algorithm

In natural environments, water drops in a river flow in a stream. Water drops overcome obstacles and barriers of the environment to find an optimum path from the source (upstream) to the destination (downstream). They flow with a particular velocity, which can be increased or decreased according to environmental factors. When water drops flow, they gain an amount of soil taken from the river bed. According to changes of the key properties (i.e. velocity and soil), water drops tend to move through the easiest path (i.e. a path with the least soil). The following subsections introduce

the fundamental elements and the procedural steps of the IWD algorithm.

3.1. Background of the IWD algorithm

The natural phenomena of water drops can be used to solve combinatorial optimization problems. In essence, the IWD algorithm is a constructive-based method, whereby a set of individuals (i.e. water drops) move in discrete steps from one node (i.e. decision variable) to the next until a complete population of solutions is reached. Water drops cooperate to change the environmental properties in order to obtain a global optimal solution. In the IWD algorithm, two key properties of natural water drops are imitated, i.e. velocity and soil, which are changed during a series of transitions pertaining to the movement of water drops. Each water drop starts with an initial velocity and zero soil. When the water drop moves from one node to another, its velocity is changed. It can also carry some soil when it moves along its path.

In the IWD algorithm, changes in the soil and velocity parameters have an influential role on the selection probability of the flow direction. The velocity is changed non-linearly, i.e., proportional to the inverse of the soil amount between two nodes in the path. Therefore, water drops in a path with less soil move faster. In each water drop, the soil amount is added non-linearly, i.e., proportional to the inverse of the time needed for the water drop to travel from its current location to the next. On the other hand, the time taken by a water drop to move from one node to another is proportional to its velocity and inversely proportional to the distance between two locations. The following sections describe the fundamental steps and mathematical modeling of the IWD algorithm.

3.2. The IWD algorithm

Fig. 4 depicts a flowchart of the main phases of the IWD algorithm. Presented as Algorithm 1, there are four main phases, i.e., initialization, solution construction, reinforcement, and termination, in the IWD algorithm. The details are as follows.

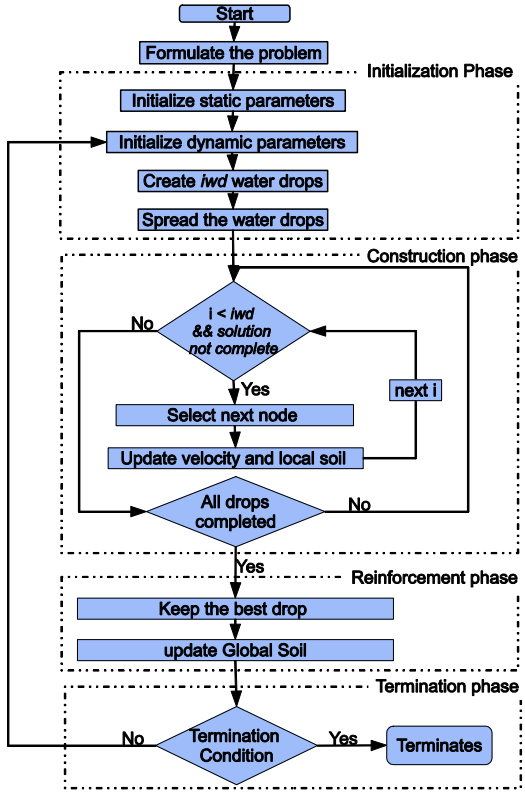

Fig. 4. A flowchart which shows the main phases of the IWD algorithm.

Algorithm 1: The main steps of the IWD algorithm

01. **Input:** Problem data set
02. **Output:** An optimal solution
03. Formulate the optimization problem as fully connected graph
04. Initialize the static parameters i.e. parameters are not changed during the search process
05. **while** Algorithm termination condition is not met **do**
06. Initialize the dynamic parameters i.e. parameters changed during the search process
07. Spread *iwd* number of IWDs randomly on a construction graph.
08. Update the list of visited node ($V_{visited}$), to include the source node
09. **while** Construction termination condition is not met **do**
10. **for** $k = 1$ to *iwd* **do**
11. i = the current node for drop k
12. j = selected next node, which does not violate problem constrains
13. move drop k from node i to node j
14. *update the following parameters*
(a) *Velocity of the drop k*
(b) Soil value within the drop k
(c) Soil value within the edge $e(ij)$
15. **End for**
16. **End while**
17. select the best solution in the iteration population (T^{IB})
18. update the soil value of all edges included in the (T^{IB})
19. update the global best solution (T^{TB})
20. if (quality of T^{TB} < quality of T^{IB})
21. $T^{TB} = T^{IB}$
22. **End while**
23. return (T^{TB})

3.2.1. Problem formulation

As shown in Algorithm 1 (line 3), the IWD algorithm uses a fully connected graph called the construction graph, i.e., $G(V,E)$, to represent an optimization problem, where $V = \{v_i | i = 1 \ldots N\}$ is a set of vertices in the graph, $E = \{(i,j) | (i,j) \in V \times V, i \neq j, i, j = 1 \ldots N\}$ is a set of edges, and N is the number of decision variables. Consider a solution, $\pi_k = \{c_{kj} | k = 1 \ldots iwd, j = 1 \ldots N\}$, where $k = 1 \ldots iwd$ represents the solution number within the population, *iwd* is the total number of solutions within the population (i.e. the number of water drops), $c_{kj} \in C$ is a set of all possible components of the solution. As an example, solution $\pi_k = \{c_{k1}, c_{k2}, c_{k3}, c_{k3}, \ldots, c_{kD}\}$, where $D \leqslant N$ is the dimension of solution k, is one of the possible permutations constructed from the possible components of C.

3.2.2. Initialization phase

As show in Algorithm 1 (line 4), the initialization phase is used to initialize a set of static and dynamic parameters of the IWD algorithm. Thereafter, the water drops are spread randomly.

3.2.2.1. Static parameters. The static parameters are initialized with static values, and they remain unchanged during the search process. They are:

- *Iwd:* is the number of water drops, which denotes a set of agents that form the solution population.
- *Velocity updating parameters* (a_v, b_v c_v): a set of parameters used to control the velocity update function, as defined in Eq. (3).
- *Soil updating parameters* (a_s, b_s, c_s): a set of parameters used to control the soil update function, as defined in Eq. (8)

- *Max_iter:* the maximum number of iterations before terminating the IWD algorithm.
- *initSoil:* the initial value of the local soil.

3.2.2.2. Dynamic parameters. The dynamic parameters are initialized at the beginning of the search, and are updated during the search process. They are reverted to their initial values at the beginning of each iteration. The dynamic parameters are:

- $V^k_{visited}$: a list of nodes visited by water drop k.
- $InitVel^k$: the initial velocity of water drop k
- $Soil^k$: the initial soil loaded on water drop k.

As an example, if water drop k starts with an empty list of visited node ($V^k_{visited}$), and with 100 and 4 for the initial soil ($InitVel^k$) and velocity ($Soil^k$) settings, respectively. The values of $V^k_{visited}$, $InitVel^k$, $Soil^k$ will be reset to empty, 100, and 4, respectively, at the beginning of each iteration

At the beginning, water drops are spread randomly at the nodes of the construction graph, and $V^k_{visited}$ is updated to include the start node.

3.2.3. Solution construction phase

The main aim of this phase is to construct a population of *iwd* solutions. A solution comprises a finite set of components, $\pi_k = \{c_{kj} | k = 1 \ldots iwd, j = 1 \ldots |D_k|\}$, D_k is the dimension of solution k. Water drop k starts with an empty set of solution components, $\pi_k = \{\}$. The first node of the tour is added to π_k whenever the water drop is spread. Then, at each step of the construction phase, the water drop extends the partial solution by traversing a new node, i.e. a feasible component that does not violate any constraints of the problem. The construction phase is completed by the transition of all water drops through the graph until the stopping criteria for constructing a complete population is met (see Algorithm 1 line 9–16). The construction phase is composed of the following steps:

3.2.3.1. Edge selection mechanism. Consider water drop k residing at the current node (node i) intends to move to the next node (node j) through an edge, $e(i,j)$, where $e \in E$. The probability of selecting $e(i,j)$ is determined by $p^k_i(j)$, as defined in Eq. (2). Then, the water drop visits node j by adding it to $V^k_{visited}$.

$$p^k_i(j) = \frac{f(soil(i,j))}{\sum_{\forall l \notin V^k_{visited}} f(soil(i,l))} \tag{2}$$

$$f(soil(i,j)) = \frac{1}{\varepsilon + g(soil(i,j))} \tag{3}$$

where ε is a small positive number used to prevent the division by zero in function $f(\cdot)$,

$$g(soil(i,j)) = \begin{cases} soil(i,j) & \text{if } min_{\forall l \notin V^k_{visited}}(soil(i,l)) \geqslant 0 \\ soil(i,j) - min_{\forall l \notin V^k_{visited}}(soil(i,l)) & \text{otherwise} \end{cases} \tag{4}$$

where $soil(i,j)$ refers to the amount of soil within the local path between nodes i and j.

3.2.3.2. Velocity and local soil Update. The velocity of water drop k at time $t+1$ is denoted by $vel^k(t+1)$. It is updated every time it moves from node i to node j using

$$vel^k(t+1) = vel^k(t) + \frac{a_v}{b_v + c_v * soil(i,j)} \tag{5}$$

where a_v, b_v, c_v are the static parameters used to represent the nonlinear relationship between the velocity of water drop k, i.e. vel^k, and the inverse of the amount of soil in the local path, i.e. $soil(i,j)$. When water drop k moves from node i to node j, both $soil^k$ (i.e. the soil within water drop k) and $soil(i,j)$ are updated using Eqs. (6) and (7), respectively.

$$soil^k = soil^k + \Delta soil(i,j) \tag{6}$$

$$soil(i,j) = (1 - \rho_n) * soil(i,j) - \rho_n * \Delta soil(i,j) \tag{7}$$

where ρ_n is a small positive constant between zero and one, $\Delta soil(i,j)$ is the amount of soil removed from the local path and carried by a water drop. Note that $\Delta soil(i,j)$ is non-linearly proportional to the inverse of vel^k, as in Eq. (8)

$$\Delta soil(i,j) = \frac{a_s}{b_s + c_s * time(i,j:vel^k(t+1))} \tag{8}$$

where, a_s, b_s, c_s are the static parameters used to represent the nonlinear relationship between $\Delta soil(i,j)$ and the inverse of vel^k. Note that $time(i,j{:}vel^k(t+1))$ refers to the time needed for water drop k to transit from node i to node j at time $t+1$. It is defined as follows.

$$time(i,j:vel^k(t+1)) = \frac{\mathrm{HUD}(i,j)}{vel^k(t+1))} \tag{9}$$

where HUD (i,j) is a heuristic desirability degree of the edge between nodes i and j.

The processes of selecting nodes to be visited as well as updating the velocity and local soil are iterated subject to the stopping criteria for obtaining a complete solution.

3.2.4. Reinforcement phase

As shown in Algorithm 1 (line 17–21), the fittest solution of each population is known as the iteration-best solution, and is denoted as T^{IB}. It is determined using Eq. (10).

$$T^{IB} = \arg\min \backslash \max_{\forall x \in T^{IWB}} q(x) \tag{10}$$

where $q(\cdot)$ is the fitness function which is used to evaluate the quality of the solutions, and T^{IWB} is the population of the solutions. To reinforce water drops in the subsequent iterations to follow T^{IB} and achieve the fittest solution over the iterations, the soil of all edges in T^{IB} is updated using Eq. (11). This update process is known as global soil update.

$$soil(i,j) = (1 + \rho_{IWD}) * soil(i,j) - \rho_{IWD} * soil^k_{IB} * \frac{1}{q(T^{IB})} \tag{11}$$

where ρ_{IWD} is a positive constant.

In each iteration, the best solution (global best), i.e., T^{TB}, is either replaced by T^{TB} or maintained, as defined in Eq. (12)

$$T^{TB} = \begin{cases} T^{IB} & \text{if } q(T^{IB}) < q(T^{TB}) \\ T^{TB} & \text{otherwise} \end{cases} \tag{12}$$

3.2.5. Termination phase

The solution construction and reinforcement phases, shown in Algorithm 1 (line 5–21), are iterated until a termination condition is met. As an example, the maximum number of iterations (i.e. *MaxIter*) is a straightforward termination condition of the IWD algorithm.

4. Proposed modifications

As stated earlier, the original IWD algorithm (Shah-Hosseini, 2007) uses the FPS method to identify the next node as a component of the constructed solution (see Fig. 4). This selection method has a number of limitations (as explained in Section 1), i.e. inability to accommodate negative soil values, sensitivity to the scaling

function, selection domination, and inability to handle indistinguishable fitness. These limitations are likely to occur with the FPS method, owing to the process of scaling the negative scale values and the behavior of the local soil updates. In this paper, we propose two ranking-based selection methods, i.e., linear ranking selection and exponential ranking selection, to overcome the limitations of the FPS method. These selection methods are adopted as part of the solution construction phase of the IWD algorithm.

4.1. Ranking selection

Ranking selection methods are based on the rationale of fitness rank, rather than on the absolute fitness value (of the soil), to determine the probability of selecting the next node. To select the next node to be included in the path, firstly, all possible arcs are ranked according to their soil values. An arc with a higher fitness (i.e. a lower soil value) is assigned a higher rank, and verse versa. All ranks are then mapped to the selection probability by a mapping function. The performance of this selection method depends on both the mapping function and the selection pressure (SP) parameter. Parameter SP refers to the tendency of selecting the best node (Al-Betar et al., 2012, 2013). Based on the mapping function used, ranking selection methods can be classified into two categories i.e. linear and non-linear selection methods. The probability of selection is determined in advance, and it remains the same throughout the search process. Two ranking selection methods, i.e., linear ranking and exponential ranking, are proposed, as follows.

4.1.1. Linear ranking selection

The linear ranking selection method uses a linear function to map a node's rank (based on its soil values) to the selection probability. Eq. (13) shows a linear function, which is introduced in Goldberg and Deb (1991).

$$P(i) = \frac{1}{N} \times \left(SP - 2(SP-1) \times \frac{i-1}{N-1} \right) \tag{13}$$

where $i \in \{1, \ldots, N\}$ represents an arc's rank. All arcs are ranked in a descending order according to their soil values, i.e., rank 1 is assigned to the fittest arc (i.e. the arc with the lowest soil value) and rank N for the least-fit arc. Parameter SP, $1 \leqslant SP \leqslant 2$, is used to control the gradient of the linear selection function. A larger SP setting results in a higher selection pressure for selecting the best node to extend the partial solution. Fig. 5 depicts the linear probability distributions of 10 individuals with different settings of SP.

4.1.2. Exponential ranking selection

Exponential ranking selection differs from linear ranking selection in that the probabilities of the ranked paths are exponentially weighted. The probabilities are governed by an exponential function, which is introduced in Goldberg and Deb (1991), as follows

$$P(i) = SP^{i-1} * \frac{1-SP}{1-SP^N} \tag{14}$$

where $i \in \{1, \ldots, N\}$ represents an arc's rank. Again, rank 1 is assigned to the fittest arc and rank N to the least-fit arc. Similarly, parameter SP, $0 < SP < 1$, is used to control the gradient of the exponential selection function. A smaller SP setting leads to a higher selection pressure of the fitter node. Fig. 6 depicts the exponential probability distributions of 10 individuals with different settings of SP.

5. Case studies

To evaluate the effectiveness of the proposed ranking selection methods, a series of experimental studies using three combinatorial optimization problems, i.e., rough set feature subset selection, multiple knapsack and travelling salesman problems, was conducted. The original IWD algorithm with the FPS method was denoted as FPS-IWD, whiles the two ranking selection methods, i.e. linear and exponential were denoted LRS-IWD and ERS-IWD, respectively. All three IWD algorithms (FPS-IWD, LRS-IWD, and ERS-IWE) were implemented using the Java programming language. The experiments were conducted using an Intel Pentium 4 core 2 Quad 2.66 GHz personal computer.

5.1. Rough set feature subset selection (RSFS)

The performance of machine learning models is affected by the dimension of the feature space of a data set, i.e. the curse-of-dimensionality problem (Yu & Liu, 2004). As such, feature selection is employed to reduce the dimension by searching the minimum subset of features that represents the meaning of the original data set (Dash & Liu, 1997). The main aim is to reduce the number of features by removing irrelevant, redundant, or noisy features. Feature selection can enhance the performance of machine learning

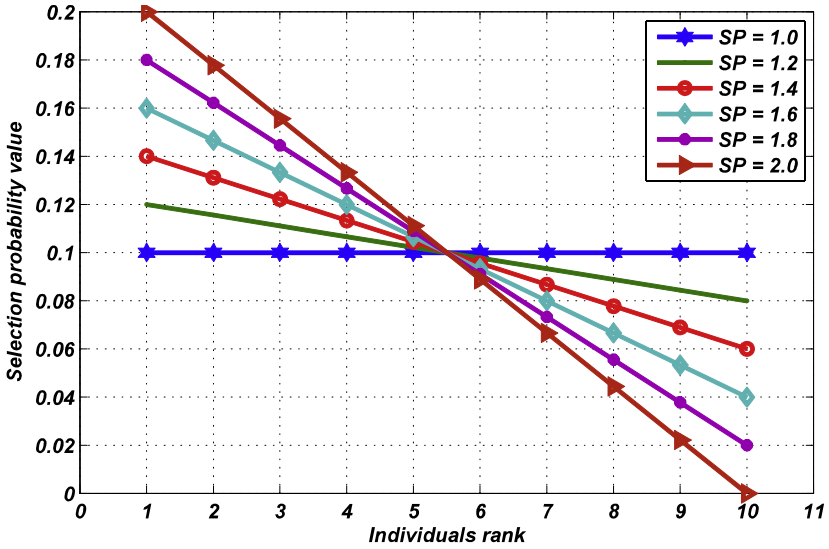

Fig. 5. Linear probability distribution of 10 individuals with different settings of SP.

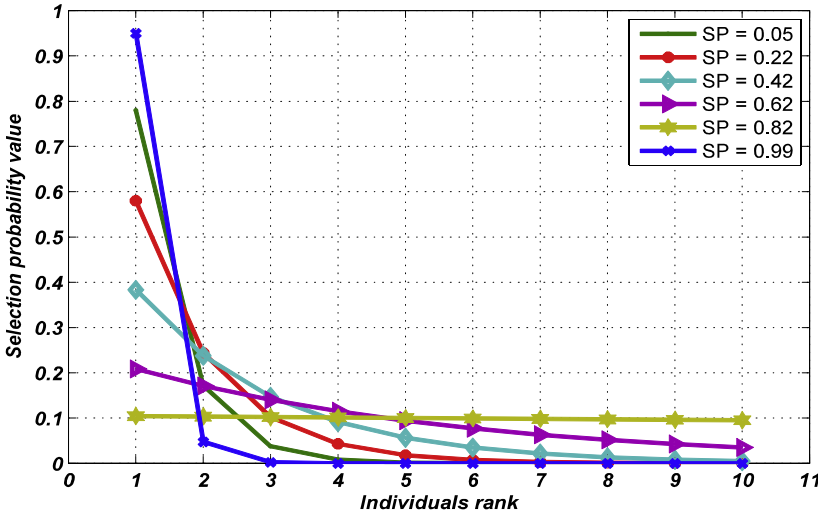

Fig. 6. Exponential probability distribution of 10 individuals with different settings of *SP*.

models in various applications, such as accelerating the data mining process and improving prediction accuracy of classifiers. Feature selection methods can be broadly classified into two types: filter-based and wrapper-based (Jensen & Shen, 2008). Filter-based methods select a subset of features using an independent evaluation function, rather than a learning algorithm (predicator). They are computationally less intensive, as compared with wrapper-based methods that integrate a learning algorithm as a black box to evaluate the subset of features. However, wrapper-based methods can be more efficient, e.g. the selected feature subset provides better classification accuracy as compared with that from filter-based methods because it directly utilizes a learning algorithm (Jensen & Shen, 2008).

Feature selection algorithms mainly include two fundamental components: subset generation and subset evaluation. Subset generation, which comprises the search algorithm, explores the problem search space for the optimal subset. Subset evaluation assesses the goodness of the generated feature subsets. In this domain, the Rough Set Theory (RST) is one of the important theories for tackling feature selection problems.

5.1.1. Rough Set Theory and rough sets

RST is a mathematical theory introduced by Pawlak (1982). It is used to analyze imperfection, i.e., the concept of indiscernibility to approximate imperfect knowledge with a pair of precise sets called the lower approximation and the upper approximation. The lower approximation is a set that includes objects definitely belong to the subset of interest. The upper approximation is a set that includes objects possibly belong to the subset of interest. A tuple of the lower approximation and the upper approximation is known as a rough set (RS) (Pawlak, 1991).

Let IS = (U, A, D) be an information system, where U is a non-empty finite set of objects (universe), D is a non-empty finite set of decision features, and A is a non-empty finite set of conditional features such that $\propto: U \rightarrow V_{\alpha}$ for every $\propto \in A$, where V_{α} is the set of values that attribute $\propto$ can take. Each non-empty $S \subseteq A$ determines an indiscernibility relationship denoted as R_S, which is defined in Eq. (15).

$$R_S = \{(x,y) \in U \times U | \forall \alpha \epsilon S, \alpha(x) = \alpha(y)\} \tag{15}$$

Note that R_S partitions U into some equivalence classes, as defined in Eq. (16).

$$U/R_S = \{[x]_S | x \in U\} \tag{16}$$

where $[X]_S$ denotes the equivalence class determined by x with respect to S, i.e.,

$$[x]_S = \{y \in U | (x,y) \in R_S\} \tag{17}$$

For simplicity, U/R_S is replaced by U/S, since R is already known.

Let $X \subseteq U$ be a target equivalence class (concept) induced by a decision feature, D. The task is to classify X by using information of equivalence classes induced by a subset, S. In general, this cannot be expressed exactly because X may include an object that is not in $[X]_s$, or vice versa. Therefore, information in X can be approximated by the lower and upper approximations of X with respect to S, which are denoted as $\underline{S}X$, and $\overline{S}X$ respectively, as defined in Eqs. (18) and (19).

$$\underline{S}X = \{x | [x]_s \subseteq X\} \tag{18}$$

$$\overline{S}X = \{x | [x]_s \cap X \neq \varnothing\} \tag{19}$$

Different regions, which include as positive ($POS(D)_s$), negative ($NEG(D)_s$), and boundary ($BND(D)_s$) regions are defined, as in Eqs. (20)–(22) respectively.

$$POS(D)_s = \bigcup_{X \in U/D} \underline{S}X \tag{20}$$

$$NEG(D)_s = U - \bigcup_{X \in U/D} \overline{S}X \tag{21}$$

$$BND(D)_s = \bigcup_{X \in U/D} \overline{S}X - \bigcup_{X \in U/D} \underline{S}X \tag{22}$$

The degree of dependency between subsets S and D is defined in Eq. (23).

$$K = \gamma_s(D) = \frac{|POS_s(D)|}{|U|} \tag{23}$$

Note that D is fully dependent on S, i.e., $D \Rightarrow S$, if and only if $k = 1$. Otherwise, D is partially dependent on S with the degree of k, i.e., $D \overset{k}{\Rightarrow} S$.

A reduct, R_{min}, is, defined as the minimal subset, R, of the original set, C, and the set of decision features, D, that has the maximum degree of dependency, $\gamma_R(D) \leqslant \gamma_C(D)$.

5.1.2. Rough set for subset feature selection (RSFS)

Rough set dependency, $\gamma_C(D)$, is a monotonic measure, i.e. if T and S are feature subsets, and $T \subseteq S$, then $\gamma_T(D) \leqslant \gamma_S(D)$. Therefore,

the main idea of RSFS is to search for the minimum subset with the degree of dependency equals to that of the full feature set. Many RS algorithms have been developed (Jensen & Shen, 2008). The simplest solution of feature selection is to generate all possible combinations, and choose the one with the minimum cardinality (Bazan & et al., 2000). Obviously, this involves an exhaustive search, and is impractical for large data sets because generating all combinations is an NP-hard problem. Consider a data set with N features. The search space contains 2^N possible feature subsets (Yu & Liu, 2004). To manage the complexity of the search process, several stochastic optimization methods, such as hill-climbing with forward selection and backward elimination (Vafaie & Imam, 1994) and meta-heuristic methods such as genetic algorithm (Vafaie & Imam, 1994), particle swarm optimization(Bae et al., 2010; Wang et al., 2007), ant colony optimization (Jensen & Shen, 2003; Zaiyadi & Baharudin, 2010), as well as great deluge and non-linear great deluge (Abdullah & Jaddi, 2010; Mafarja & Abdullah, 2011) can be used. In the following section, an experimental study using the modified IWD algorithm for feature selection with RS is presented.

5.1.3. Experimental study

To evaluate FPS-IWD, LRS-IWD, and ERS-IWD, a series of experiments pertaining to RSFS problems were conducted. Specifically, a total of 13 benchmark data sets from Jensen and Shen (2003) were experimented. Most of the data sets are UCI benchmark data sets (Bache & Lichman, 2013). Table 1 summarizes the characteristics of the data sets. The complexity (dimension) of the data sets ranges from low (13) to high (56). Before commencing the experiment, the necessary pre-processing steps pertaining to the data sets were conducted, i.e. discretizing real-valued features, treating missing values, and removing outlier instances (Jensen & Shen, 2001).

Different optimization problem has different parameter settings. For tackling the RSFS problem, the static and dynamic parameters of the original IWD algorithm were set to the values listed in Table 2, which were used in Shah-Hosseini (2009), in order to facilitate performance comparison with the results of FPS-IWD reported therein. The number of water drops (*iwd*) was set to the number of features in the data set. Note that ε_s is a small positive used to prevent the division by zero in Eq. (3). Accordingly, ε_s has no significant impact to the performance of the IWD algorithm. It has been omitted in both ERS-IWD and LRS-IWD. The other parameters are maintained the same for all FPS-IWD, ERS-IWD, and LRS-IWD.

Table 1
Characteristics of the data sets used for RSFS.

Data-sets	No. of features	No. of instances
M-of-N	13	1000
EXACTLY	13	1000
EXACTLY2	13	1000
HEART	13	270
VOTE	16	300
CREDIT	20	1000
MUSHROOM	22	8124
LED	24	2000
LETTERS	25	26
DERM	34	366
DERM2	34	366
WQ	38	521
LUNG	56	32

Table 2
Parameters settings of the IWD algorithm for RSFS.

Type	Parameters	Values
Static	$_{IWD}$	Number of features
	a_v, b_v, c_v	1, 0.01, 1
	a_s, b_s, c_s	1, 0.01, 1
	initSoil	1000
	Max_iter	250
	ε_s, ρ_{IWD}, ρ_n	0.01, 0.9,0.9
Dynamic	$V_c^{IWD_k}$	Empty
	$initVel^{IWDk}$	4
	$soil^{IWDk}$	0

An appropriate SP setting is critical in ensuring adequate optimization progress on one hand and allowing escape from the local optima on the other hand. To investigate the influence of *SP* on the performance of ERS-IWD and LRS-IWD, a preliminary experiment was conducted with three data sets (i.e., M-of-N, Heart, and LED) to fine-tune the *SP* value. For the ERS-IWD algorithm, six *SP* settings ranging from a low to a high value, i.e., *SP* = (0.22, 0.42, 0.62, 0.82, 0.9, 0.99), were evaluated. For each data set, the experiment was executed 20 runs, as the IWD algorithm could produce a different result in each run owing to its stochastic nature. The overall results are summarized in Table 3. Each result is presented as a number referring to the reduct length, and its superscript denotes the frequency of the reduct length from 20 runs. Fig. 7 shows that, in general, increasing *SP* leads to improving the performance (average length of 20 reducts) of ERS-IWD. The results (Table 3) indicated that the optimal *SP* setting for ERS-IWD was 0.42, which produced the optimal reducts for the three data sets from 20 runs. In general, increasing SP drives ERS-IWD towards exploitation rather exploration. As such, ERS-IWD is likely to be trapped in local optima.

For LRS-IWD, five different *SP* settings ranging from a low to a high value, i.e. *SP* = (1.2, 1.4, 1.6, 1.8, 1.99), were tested. The results are summarized in Table 4. The best *SP* setting of LRS-IWD was 1.8, which produced best results for the three data sets. Fig. 8 shows that increasing *SP* leads to improving the performance of LRS-IWD. However, when SP reached 1.99, the results deteriorated.

From the preliminary experimental results with different *SP* settings, it could be observed that the landscapes of the solution space did not require a deep search to reach the optimal solutions.

A thorough evaluation with 13 data sets, using *SP* = 0.42 and *SP* = 1.8 for both ERS-IWD and LRS-IWD respectively, was conducted. For performance comparison, the results of FPS-IWD reported in Alija and et al. (2013) were used. Besides that, the results of three optimization methods, i.e., greedy hill-climbing (RSAR), genetic algorithm (GenRSAR), and ant colony (AntRSAR), reported in Jensen and Shen (2003), were also used for comparison purposes. Table 5 shows the overall results.

It can be seen that RSAR produced the same reducts from 20 runs for all 13 data sets. In other words, RSAR provided only a single solution. The other algorithms could produce reducts with different lengths. For most of the data sets, RSAR and GenRSAR did not manage to produce reducts with the optimal length, as compared with those from AntRSAR and the IWD algorithm (FPS-IWD, ERS-IWD, and LRS-IWD). FPS-IWD did not provide the absolute optimal reducts (i.e. the same reducts length from 20 runs) for any of the 13 data sets, while ERS-IWD and LRS-IWD yielded the absolute optimal reducts for 6 and 7 data sets, respectively. Furthermore, both of ERS-IWD and LRS-IWD performed more stably as compared with FPS-IWD.

The performances of ERS-IWD and LRS-IWD were further evaluated using the *t*-test. The p-values of the *t*-test for both ERS-IWD and LRS-IWD are shown in Table 6. The results indicate that, for all data sets, there is no significant difference in performance between both methods. The *t*-test results between ERS-IWD and FPS-IWD are summarized in Table 7. It can be observed that there is a significant difference in the results from most of the data sets. The p-values of four data sets (i.e. M-of-N, Exactly2, LETTERS, and WQ) show no significant difference in performance for both methods.

Table 3
Comparison of the results of the ERS-IWD algorithm with various *SP* values.

Data set	*SP* = 0.99	*SP* = 0.9	*SP* = 0.82	*SP* = 0.62	*SP* = 0.42	*SP* = 0.22
M-of-N	$6^{(1)}7^{(12)}\ 8^{(7)}$	$6^{(2)}7^{(18)}$	$6^{(14)}7^{(6)}$	$6^{(19)}7^{(1)}$	$6^{(20)}$	$6^{(19)}7^{(1)}$
Heart	$7^{(16)}\ 8^{(4)}$	$6^{(2)}7^{(18)}$	$6^{(13)}7^{(7)}$	$6^{(20)}$	$6^{(20)}$	$6^{(20)}$
LED	$11^{(3)}12^{(6)}\ 13^{(4)}\ 14^{(3)}\ 15^{(4)}$	$8^{(2)}\ 9^{(3)}\ 10^{(2)}\ 11^{(2)}\ 12^{(4)}\ 13^{(3)}\ 14^{(3)}\ 15^{(1)}$	$6^{(1)}7^{(2)}8^{(6)}\ 9^{(8)}\ 10^{(2)}\ 11^{(1)}$	$5^{(14)}6^{(6)}$	$5^{(20)}$	$5^{(18)}6^{(2)}$

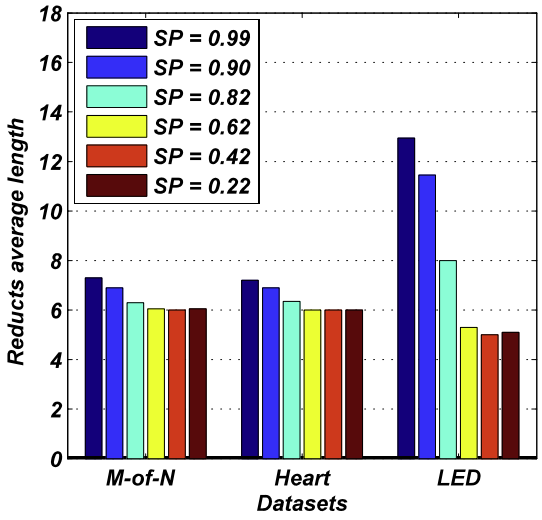

Fig. 7. Average length of reducts from 20 runs of the ERS-IWD algorithm with various *SP* values.

Table 4
Comparison of the results of the LRS-IWD algorithm with various *SP* values.

Data set	*SP* = 1.2	*SP* = 1.4	*SP* = 1.6	*SP* = 1.8	*SP* = 1.99
M-of-N	$6^{(9)}\ 7^{(9)}\ 8^{(2)}$	$6^{(14)}7^{(6)}$	$6^{(20)}$	$6^{(20)}$	$6^{(19)}7^{(1)}$
Heart	$6^{(13)}7^{(7)}$	$6^{(17)}\ 7^{(3)}$	$6^{(20)}$	$6^{(20)}$	$6^{(20)}$
LED	$5^{(4)}6^{(9)}7^{(7)}$	$5^{(7)}\ 6^{(9)}\ 7^{(4)}$	$5^{(5)}6^{(15)}$	$5^{(9)}\ 6^{(11)}$	$5^{(8)}\ 6^{(11)}\ 7^{(1)}$

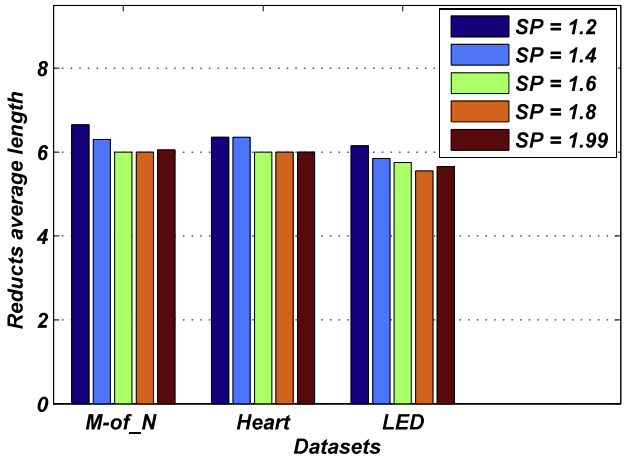

Fig. 8. Average length of reducts from 20 runs of the LRS-IWD algorithm with various *SP* values.

In general, based on the RSFS benchmark problems, the experimental results indicate that both ERS-IWD and LRS-IWD outperform FPS-IWD, RSAR, and GenRSAR, and is comparable with AntRSAR. As such, the attempt to replace FPS with the ranking based selection methods is useful, and the results are better than those from FPS-IWD.

To compare the computation time of the proposed model, the average computation time of the data set MUSHROOM (a large data set with 8124 data samples, each sample with 22 features) was considered as an example. The average computation times of FPS-IWD, ERS-IWD, and LRS-IWD were 4.635, 26.953, and 9.108 min, respectively. These results showed that the proposed ranking-based methods were computational expensive. In particular, the computational time of ERS-IWD was almost 6 times more than that required by FPS-IWD, indicating the shortcomings of the proposed modifications to the IWD algorithm.

5.2. The multiple knapsack problem (MKP)

The multiple knapsack problem (MKP) is an NP-hard combinatorial optimization problem (Chu & Beasley, 1998), that occurs in many practical situations. It is generalization of the single knapsack problem. Consider V a set of n items (variables) and K is a set of m knapsacks (constrains). Each item $i \in V$ has a profit P_i, and each knapsack $j \in V$ has a capacity C_j, and weight W_{ij} of item i in knapsack j (Azad, Rocha, & Fernandes, 2013). The main goal MKP is to find a subset $U \subseteq V$ that accords the maximum profit Z, as defined in Eq. (24), and all elements have a feasible packing solution in the knapsack, i.e. all the constrains are satisfied, as defined in Eq. (25) (the total weight of the items in the knapsack does not exceed the knapsack capacities). Formally, MKP can be presented as follows.

$$\text{maximize } Z = \sum_{j=1}^{m}\sum_{i=1}^{n} P_i x_{ij} \tag{24}$$

subject to

$$\sum_{i=1}^{n} W_{ij}x_{ij} \leqslant C_j \; j \in V = \{1,\ldots,m\} \tag{25}$$

$$x_{ij} = 0 \text{ or } 1 \quad i \in K, j \in V \tag{26}$$

$$x_{ij} = \begin{cases} 1 & \text{if item i is assigned to knapsack } j \\ 0 & \text{otherwise} \end{cases} \tag{27}$$

To solve MKP using the IWD algorithm, a complete weighted directed graph, $G = (V, E, d)$, of the problem needs to be defined, where V is a set of vertices corresponding to the items, and $E = \{(x, y)|(x, y) \in V \times V\}$ is a set of arcs that is fully connected to all the graph vertices. Arc $E(x,y)$ associates vertices x and y, which indicates that item y can be chosen after item x. In the IWD algorithm, the value of $soil(x,y)$ is associated with the arc between any two vertices x and y. The water drops are distributed randomly as the set of items. These water drops move independently from one node (i.e. item) to the next, and construct a feasible solution.

5.2.1. Experimental study

In this study, ten instances of the benchmark MKP problem taken from "mknapcb1" of the OR-Library (http://people.brunel.ac.uk/~mastjjb/jeb/orlib/) were considered. Table 8 lists the details, where the data set name, the number of items (n), and the number

Table 5
Performance comparison of the proposed methods with other algorithms on rough set feature subset selection benchmark problems.

Num	Data set	Features	RSAR (Jensen & Shen, 2003)	GenRSAR (Jensen & Shen, 2003)	AntRSAR (Jensen & Shen, 2003)	FPS-IWD (Alijla et al., 2013)	LRS-IWD	ERS-IWD
1	M-of-N	13	$8^{(20)}$	$6^{(6)}\ 7^{(12)}$	$6^{(20)}$	$6^{(18)}7^{(2)}$	$6^{(20)}$	$6^{(20)}$
2	Exactly	13	$9^{(20)}$	$6^{(10)}\ 7^{(10)}$	$6^{(20)}$	$6^{(15)}7^{(5)}$	$6^{(20)}$	$6^{(20)}$
3	Exactly2	13	$13^{(20)}$	$10^{(9)}\ 11^{(11)}$	$10^{(20)}$	$10^{(19)}12^{(1)}$	$10^{(20)}$	$10^{(20)}$
4	HEART	13	$7^{(20)}$	$6^{(18)}\ 7^{(2)}$	$6^{(18)}\ 7^{(2)}$	$7^{(3)}8^{(13)}9^{(4)}$	$6^{(20)}$	$6^{(20)}$
5	VOTE	16	$9^{(20)}$	$8^{(2)}\ 9^{(18)}$	$8^{(20)}$	$9^{(2)}10^{(9)}11^{(5)}12^{(4)}$	$8^{(10)}9^{(10)}$	$8^{(11)}9^{(9)}$
6	CREDIT	20	$9^{(20)}$	$10^{(6)}\ 11^{(14)}$	$8^{(12)}9^{(4)}10^{(4)}$	$10^{(12)}11^{(8)}$	$8^{(3)}\ 9^{(15)}\ 10^{(2)}$	$8^{(3)}\ 9^{(15)}\ 10^{(2)}$
7	MUSHROOM	22	$5^{(20)}$	$5^{(1)}\ 6^{(5)}\ 7^{(14)}$	$4^{(20)}$	$4^{(2)}5^{(5)}6^{(8)}7^{(4)}8^{(1)}$	$4^{(20)}$	$4^{(20)}$
8	LED	24	$12^{(20)}$	$6^{(1)}\ 7^{(3)}\ 8^{(16)}$	$5^{(12)}\ 6^{(4)}\ 7^{(3)}$	$7^{(6)}8^{(14)}$	$5^{(9)}6^{(11)}$	$5^{(20)}$
9	LETTERS	25	$9^{(20)}$	$8^{(8)}\ 9^{(12)}$	$8^{(20)}$	$8^{(16)}9^{(4)}$	$8^{(16)}9^{(4)}$	$8^{(13)}9^{(7)}$
10	DERM	34	$7^{(20)}$	$10^{(6)}\ 11^{(14)}$	$6^{(17)}\ 7^{(3)}$	$6^{(2)}7^{(3)}8^{(5)}9^{(5)}10^{(5)}$	$6^{(6)}7^{(14)}$	$6^{(8)}7^{(12)}$
11	DERM2	34	$10^{(20)}$	$10^{(4)}\ 11^{(16)}$	$8^{(3)}\ 9^{(17)}$	$8^{(1)}9^{(3)}10^{(6)}11^{(8)}12^{(1)}13^{(1)}$	$9^{(13)}\ 10^{(7)}$	$9^{(16)}\ 10^{(4)}$
12	WQ	38	$14^{(20)}$	16	$12^{(2)}13^{(7)}14^{(11)}$	$13^{(3)}14^{(17)}$	$14^{(17)}15^{(3)}$	$13^{(4)}14^{(15)}15^{(1)}$
13	LUNG	56	$4^{(20)}$	$6^{(8)}\ 7^{(12)}$	$4^{(20)}$	$4^{(6)}5^{(12)}6^{(2)}$	$4^{(20)}$	$4^{(20)}$

Table 6
Significance tests of the LRS-IWD and ERS-IWD algorithms using *t*-test with $\propto < 0.05$. "None" denotes that both results are the same (i.e. no valid *p*-values). Highlight (bold) denote that result is significantly different.

Num	Data set	LRS-IWD	ERS-IWD	P-value
1	M-of-N	$6^{(20)}$	$6^{(20)}$	None
2	Exactly	$6^{(20)}$	$6^{(20)}$	None
3	Exactly2	$10^{(20)}$	$10^{(20)}$	None
4	HEART	$6^{(20)}$	$6^{(20)}$	None
5	VOTE	$8^{(10)}9^{(10)}$	$8^{(11)}9^{(9)}$	0.3795069
6	CREDIT	$8^{(2)}\ 9^{(15)}10^{(3)}$	$8^{(3)}\ 9^{(15)}\ 10^{(2)}$	0.2696267
7	MUSHROOM	$4^{(20)}$	$4^{(20)}$	None
8	LED	$5^{(9)}6^{(11)}$	$5^{(20)}$	**1.167E−05**
9	LETTERS	$8^{(16)}9^{(4)}$	$8^{(13)}9^{(7)}$	0.1500967
10	DERM	$6^{(6)}7^{(14)}$	$6^{(8)}7^{(12)}$	0.2598696
11	DERM2	$9^{(13)}\ 10^{(7)}$	$9^{(16)}\ 10^{(4)}$	0.1500967
12	WQ	$14^{(17)}15^{(3)}$	$13^{(4)}14^{(15)}15^{(1)}$	0.0892385
13	LUNG	$4^{(20)}$	$4^{(20)}$	None

Table 7
Significance tests of the FPS-IWD and ERS-IWD algorithms using t-test with $\propto < 0.05$. Highlight (bold) denote that result is significantly different.

Num	Data set	FPS-IWD	ERS-IWD	P-value
1	M-of-N	$6^{(18)}7^{(2)}$	$6^{(20)}$	0.0772209
2	Exactly	$6^{(15)}7^{(5)}$	$6^{(20)}$	**0.0080948**
3	Exactly2	$10^{(19)}12^{(1)}$	$10^{(20)}$	0.161818
4	HEART	$7^{(3)}8^{(13)}9^{(4)}$	$6^{(20)}$	**5.343E−18**
5	VOTE	$9^{(2)}10^{(9)}11^{(5)}12^{(4)}$	$8^{(11)}9^{(9)}$	**6.132E−11**
6	CREDIT	$10^{(12)}11^{(8)}$	$8^{(3)}\ 9^{(15)}\ 10^{(2)}$	**5.039E−11**
7	MUSHROOM	$4^{(2)}5^{(5)}6^{(8)}7^{(4)}8^{(1)}$	$4^{(20)}$	**6.521E−10**
8	LED	$7^{(6)}8^{(14)}$	$5^{(20)}$	**6.418E−26**
9	LETTERS	$8^{(16)}9^{(4)}$	$8^{(13)}9^{(7)}$	0.150097
10	DERM	$6^{(2)}7^{(3)}8^{(5)}9^{(5)}10^{(5)}$	$6^{(8)}7^{(12)}$	**6.844E−07**
11	DERM2	$8^{(1)}9^{(3)}10^{(6)}11^{(8)}12^{(1)}13^{(1)}$	$9^{(16)}\ 10^{(4)}$	**3.978E−05**
12	WQ	$13^{(3)}14^{(17)}$	$13^{(4)}14^{(15)}15^{(1)}$	0.089238
13	LUNG	$4^{(6)}5^{(12)}6^{(2)}$	$4^{(20)}$	**5.16E−07**

Table 8
The main properties of 10 mknapcb1 instances obtained from the OR-Library.

Data set	*n*	*m*	Optimal profit
5.100–10	100	5	42757
5.100–11	100	5	42545
5.100–12	100	5	41968
5.100–13	100	5	45090
5.100–14	100	5	42218
5.100–15	100	5	42927
5.100–16	100	5	42009
5.100–17	100	5	45020
5.100–18	100	5	43441
5.100–19	100	5	44554

Table 9
Parameters settings of the IWD algorithm for MKP.

Type	Parameters	Values
Static	$_{IWD}$	30
	a_v, b_v, c_v	1, 0.01, 1
	a_s, b_s, c_s	1, 0.01, 1
	initSoil	1000
	Max_iter	200
	ε_s, ρ_{IWD},ρ_n	0.01, 0.9,0.9
Dynamic	$V_c^{IWD_k}$	Empty
	$initVel^{IWDk}$	4
	$soil^{IWDk}$	0

of constraints (m), as well as the best feasible solution profit (Optimal profit) are those reported in Chu and Beasley (1998) ("mkcbres.txt").

To facilitate performance comparison with FPS-IWD, the static and dynamic parameters were set to the those used in Shah-Hosseini (2008), as shown in Table 9. Parameter ε_s has no significant impact on the performance of FPS-IWD, and has been omitted in both ERS-IWD and LRS-IWD. After several trials, the number of water drops (*iwd*) and the maximum number of iterations (*Max_iter*) were set to the values listed in Table 9. For the proposed ranking-based methods, i.e. ERS-IWD, and LRS-IWD, a series of preliminary experiments was conducted using the first three data sets, in order to determine the optimal *SP* settings. Each experiment was repeated 10 times.

Table 10
Average profits from 10 runs of the ERS-IWD method with different SP values. The highlighted (bold) values denote the best result of each data set.

Data instance	*SP* = 0.02	*SP* = 0.22	*SP* = 0.42	*SP* = 0.62	*SP* = 0.82	*SP* = 0.99
5.100–10	42488.4	42470.9	42469.7	**42673.5**	42576.9	42511.6
5.100–11	42085.7	42104.3	42113.7	**42380.8**	42244.5	42142.2
5.100–12	41239.3	41256.7	41317.8	**41607.5**	41442.2	41331.2

Table 11
Average profits from 10 runs of the LRS-IWD method with different SP values. The highlighted (bold) values denote the best result of each data set.

Data instance	*SP* = 1.2	*SP* = 1.4	*SP* = 1.6	*SP* = 1.8	*SP* = 1.9	*SP* = 1.99
5.100–10	42620.1	42625.1	42646.4	42628.7	**42650.8**	42635.7
5.100–11	42335.3	42336.8	42346.9	42337.7	**42347.2**	42330.8
5.100–12	41533.6	41531.3	41534.3	41517	**41545.6**	41530.3

Tables 10 and 11, respectively, show the average profits for both ERS-IWD and LRS-IWD with different *SP* settings. In general, increasing SP in LRS-IWD (i.e. larger *SP*) and decreasing SP in ERS-IWD (i.e. larger *SP* value) significantly improved their performances. Accordingly, $SP = 0.62$ and $SP = 1.9$ were set for ERS-IWD and LRS-IWD, respectively. The results implied that the search landscape of the MKP problem was flat, and more exploration than exploitation was required.

A thorough evaluation was conducted with ten data sets using $SP = 0.62$ and $SP = 1.9$ for both ERS-IWD and LRS-IWD, respectively. We repeated the IWD algorithm 10 times for each data set. Table 12 compares the results from FPS-IWD, ERS-IWD, and LRS-IWD, whereby the best, worst, average, and standard deviation (STD) of the 10 runs are presented. In addition, the best known optimal values for the data sets, as reported in Azad et al. (2013), are presented. Note that the results of FPS-IWD are obtained by running the original IWD algorithm in Shah-Hosseini (2008).

The results in Table 12 indicate that ERS-IWD outperformed LRS-IWD and FPS-IWD, while LRS-IWD performed better than FPS-IWD in most of the data sets. The significance difference of the results was tested using the hypothesis test (i.e. *t*-test). The *t*-test results, over the 10 data sets, show that ERS-IWD significantly outperformed both of LRS-IWD and FPS-IWD, with *p*-values of 7.27E−04 and 8.99E−06 respectively. The results of LRS-IWD are significantly better than those from FPS-IWD, with a p-value of 1.32E−05.

To compare the computation time of the proposed methods, the average computation time of the 5.100–10 data set was considered as an example (note that all data sets followed the same numbers of items and constraints). The average computation times of FPS-IWD, ERS-IWD, and LRS-IWD were 0.323, 0.372, and 0.343 min, respectively. The computational requirements of ERS-IWD and LRS-IWD were higher than that of FPS-IWD. The computational time of ERS-IWD was 1.15 times more than that of FPS-IWD.

5.3. The travelling salesman problem (TSP)

TSP is a well-known NP-hard combinatorial optimization problem. The main aim of TSP is to find the shortest route of a salesman, who is required to visit a given number of cities, each exactly once, and to return to the starting city. With N cities, there are $(N-1)!/2$ feasible solutions, in which the global optimum solution is sought. TSP can be formulated as a complete weighted directed graph, $G = (V, E, d)$, where V is a set of vertices (cities), $E = \{(i,j)|(i,j) \in V \times V\}$ is a set of edges associated with a pair of vertices i, and j, d: $E \rightarrow N$ is a weighting function linking a positive number as a weight/cost d_{ij} with the edge between vertices i and j. The main goal of TSP is to find a tour (i.e. a Hamiltonian cycle in a city graph) that has the minimum cost.

To solve TSP using the IWD algorithm, $soil(i,j)$ is associated with each edge between two vertices i and j. The water drops reside randomly in the set of vertices (cities), where each water drop has an initial value of soil and velocity. The water drops move independently from one vertex to the next, and construct a feasible solution (a route from the source to the destination).

5.3.1. Experimental study

A total of eight benchmark TSP data sets were experimented, as shown in the first column of Table 16. The data sets were obtained from TSPLIB (Reinelt, 1991). The chosen data sets exhibit varying degrees of difficulties, i.e. from 51 to 318 cities (i.e. the number associated with each name denotes the number of cities). The

Table 12
Performance comparison of ERS-IWD, LRS-IWD, and FPS-IWD in solving 10 instances of mknapcb1 data set obtained from the OR-Library. The highlighted (bold) values denote the best result for each instance.

Data set	IWD algorithm	Optimal profit	Best	Worst	Average	Deviation (%)	STD
5.100–10	FPS-IWD	42757	42582	42465	42500.1	0.600837	56.51637
	ERS-IWD		**42705**	**42670**	**42673.5**	**0.19529**	**11.07**
	LRS-IWD		42702	42643	42671.5	0.199967	24.12122
5.100–11	FPS-IWD	42545	42179	42044	42108.9	1.025032	54.30664
	ERS-IWD		**42418**	**42345**	**42376**	**0.397226**	**24.56**
	LRS-IWD		42445	42345	42365.3	0.422376	37.97
5.100–12	FPS-IWD	41968	41420	41168	41210.4	1.805185	77.40543
	ERS-IWD		**41694**	**41547**	**41607.5**	**0.858988**	**45.17**
	LRS-IWD		41672	41525	41552.5	0.99004	53.76
5.100–13	FPS-IWD	45090	44070	43907	43996.9	2.424263	66.15042
	ERS-IWD		**44507**	**44358**	**44422**	**1.481481**	**46.88**
	LRS-IWD		44553	44320	44381.6	1.57108	71.72
5.100–14	FPS-IWD	42218	42025	41794	41855.6	0.858402	81.15171
	ERS-IWD		**42085**	**42047**	**42051.9**	**0.393434**	**11.77**
	LRS-IWD		42144	42035	42051.8	0.393671	33.78
5.100–15	FPS-IWD	42927	42648	42648	42648	0.649941	0
	ERS-IWD		**42831**	**42781**	**42797.3**	**0.302141**	**19.61**
	LRS-IWD		42787	42699	42739.8	0.436089	30.61
5.100–16	FPS-IWD	42009	41554	41395	41495.7	1.221881	61.62981
	ERS-IWD		**41904**	**41820**	**41832.2**	**0.420862**	**26.44**
	LRS-IWD		41900	41756	41800.3	0.496798	40.33
5.100–17	FPS-IWD	45020	44419	44308	44331.1	1.530209	46.41228
	ERS-IWD		**44881**	**44836**	**44850.7**	**0.376055**	**13.99**
	LRS-IWD		44839	44750	44808.6	0.469569	30.21
5.100–18	FPS-IWD	43441	43134	42957	42993.1	1.031054	53.68933
	ERS-IWD		**43381**	**43151**	**43217.1**	**0.515412**	**66.91**
	LRS-IWD		43317	43146	43183.8	0.592067	51.04
5.100–19	FPS-IWD	44554	44342	44164	44249.8	0.682767	86.43533
	ERS-IWD		**44529**	**44433**	**44470.6**	**0.187189**	**35.01**
	LRS-IWD		44472	44411	44450.4	0.232527	23.19

Table 13
Parameters setting of the IWD algorithm for TSP.

Type	Parameters	Values
Static	$_{IWD}$	70
	a_v, b_v, c_v	1, 0.01, 1
	a_s, b_s, c_s	1, 0.01, 1
	initSoil	1000
	Max_iter	7000
	ε_s, ρ_{IWD},ρ_n	0.01, 0.9,0.9
Dynamic	V_c^k	Empty
	$initVel^k$	100
	$soil^k$	0

Table 14
The average tour lengths from 10 runs of the ERS-IWD algorithm with different *SP* values. The highlighted (bold) values denote the best result for each data set.

Data set	*SP* = 0.02	*SP* = 0.2	*SP* = 0.4	*SP* = 0.6	*SP* = 0.8	*SP* = 0.99
eil51	**440.3**	477.1	519.2	564.0	631.2	670.4
eil76	**551.8**	646.4	706.0	796.4	909.3	1007.2
eil101	**671.3**	792.8	873.3	987.0	1179.7	1360.0

Table 15
The average tour lengths from 10 runs of the LRS-IWD algorithm with different *SP* values. The highlighted (bold) values denote the best result for each data set.

Data set	*SP* = 1.2	*SP* = 1.4	*SP* = 1.6	*SP* = 1.8	*SP* = 1.9	*SP* = 1.99
eil51	808.2	729.3	656.4	619.0	569.0	**515.7**
eil76	1141.0	1036.5	982.2	939.6	840.8	**795.5**
eil101	1552.1	1420.4	1329.0	1223.0	1141.0	**1048.9**

static and dynamic parameters were set to those summarized in Table 13, as used in Shah-Hosseini (2009), in order to facilitate comparison with FPS-IWD. After several trials, the number of water drops (*iwd*) and the maximum number of iterations (Max_iter) were set as listed in Table 13.

For all FPS-IWD, ERS-IWD, and LRS-IWD experiments, the parameter settings were maintained except ε_s, which was omitted in both ERS-IWD, and LRS-IWD after replacing the selection method. The SP parameter for both ERS-IWD and LRS-IWD was tuned empirically using three data sets, i.e., eil51, eil76, and eil101. A total of 10 runs were conducted for each TSP data set. Tables 14 and 15, respectively; show the results (average tour length from 10 runs) pertaining to the *SP* settings for both ERS-IWD and LRS-IWD. Figs. 9 and 10 depict the results of ERS-IWD and LRS-IWD graphically. In general, it can be seen that increasing SP (i.e. for ERS-IWD, a lower *SP* value indicates a higher SP setting, and the opposite for LRS-IWD) led to performance improvement. The best *SP* settings of ERS-IWD and LRS-IWD were 0.02 and 1.99, respectively. These *SP* settings, i.e. higher SP settings gave better results, for both of ERS-IWD and LRS-IWD indicated the rough landscape of the solution space for the TSP problems, which required an intensive and deep search to reach the optimal solution.

Extensive experiments with eight TSPLIB data sets using the selected *SP* settings were the carried out. Each experiment was repeated 10 runs. Table 16 summarizes the results of ERS-IWD, LRS-IWD, and FPS-IWD, whereby the best, worst, average, and standard deviation (STD) of 10 runs are presented. The best known (optimal) solution for each TSPLIB data set is presented too. In addition, the deviations (in percentage) of the average solution from the best known solution are presented. The results of FPS-IWD were obtained by carrying out the original IWD algorithm

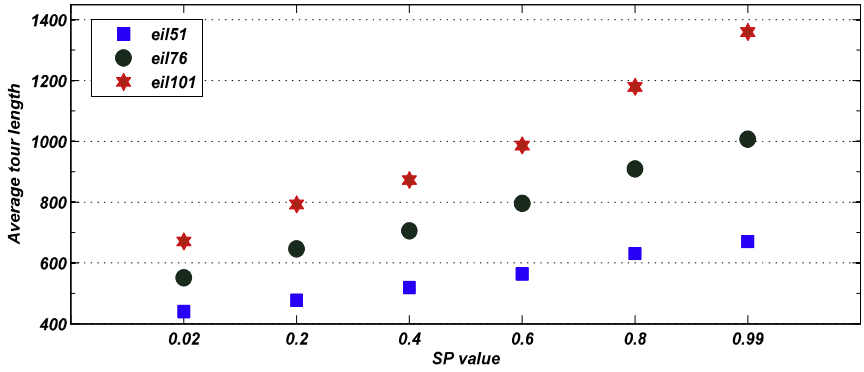

Fig. 9. Decrement of the *SP* value results in improvement of the ERS-IWD performance.

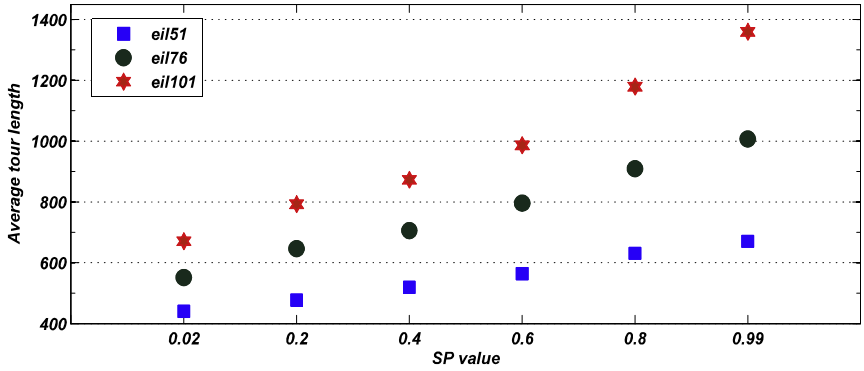

Fig. 10. Increment of the *SP* value results in improvement of the performance of the LRS-IWD algorithm.

Table 16
Compare the performance of the ERS-IWD, LRS-IWD, and FPS-IWD algorithms in solving eight TSPLIB data sets. The highlighted (bold) values denote the best result for each data set.

Data set	IWD algorithm	Best known	Best	Worst	Average	Deviation (%)	STD
eil51	FPS-IWD	**426**	448	484	470.79	10.52	11.68
	ERS-IWD		**429**	**449**	**440.30**	**3.36**	**6.50**
	LRS-IWD		497	528	515.70	21.06	9.18
eil76	FPS-IWD	**538**	597	650	617.91	14.85	16.86
	ERS-IWD		**545**	**559**	**551.80**	**2.57**	**4.73**
	LRS-IWD		772	812	795.50	47.86	11.45
eil101	FPS-IWD	**629**	726	797	752.16	19.58	26.12
	ERS-IWD		**654**	**682**	**671.30**	**6.72**	**8.38**
	LRS-IWD		1017	1063	1048.90	66.76	14.84
ch130	FPS-IWD	**6110**	6606	7047	6820.03	11.62	136.48
	ERS-IWD		**6316**	**6461**	**6403.40**	**4.80**	**41.66**
	LRS-IWD		12853	13370	13121.00	114.75	150.47
kroA100	FPS-IWD	**21282**	23706	24824	24162.10	13.53	415.09
	ERS-IWD		**21959**	**22786**	**22298.50**	**4.78**	**273.24**
	LRS-IWD		43436	44739	44104.90	107.24	443.27
kroA200	FPS-IWD	**29368**	35794	38584	37316.04	27.06	895.81
	ERS-IWD		**31680**	**32372**	**32032.50**	**9.07**	**231.17**
	LRS-IWD		85010	87054	86038.30	192.97	621.71
lin105	FPS-IWD	**14379**	15341	15984	15528.00	7.99	192.69
	ERS-IWD		**14696**	**14908**	**14801.80**	**2.94**	**71.23**
	LRS-IWD		28979	30845	29878.00	107.79	503.86
lin318	FPS-IWD	**42029**	58047	61221	59324.57	41.15	896.62
	ERS-IWD		**47021**	**48046**	**47481.10**	**12.97**	**297.18**
	LRS-IWD		152419	159267	156421.10	272.17	2314.76

Table 17
Significance tests of the ERS-IWD and LRS-IWD algorithms using t-test with $\propto < 0.05$. Highlight (bold) denote that result is significantly different.

Data set	ERS-IWD	LRS-IWD	P-value
eil51	440.3	515.7	**1.75E−14**
eil76	551.8	795.5	**9.14E−23**
eil101	671.3	1048.9	**1.08E−23**
ch130	6403.4	13121	**7.15E−29**
kroA100	22298.5	44104.9	**1.16E−28**
kroA200	32032.5	86038.3	**7.42E−34**
lin105	14801.8	29878	**5.84E−26**
lin318	47481.1	156421.1	**1.65E−29**

Table 18
Significance tests of the FPS-IWD and LRS-IWD algorithms using t-test with $\propto < 0.05$. Highlight (bold) denote that result is significantly different.

Data set	FPS-IWD	LRS-IWD	P-value
eil51	470.79	515.7	**8.90E−09**
eil76	617.91	795.5	**1.8E−16**
eil101	752.16	1048.9	**1.96E−17**
ch130	6820.03	13121	**2.56E−26**
kroA100	24162.10	44104.9	**9.19E−27**
kroA200	37316.04	86038.3	**3.62E−29**
lin105	15528.00	29878	**4.05E−25**
lin318	59324.57	156421.1	**3.96E−28**

Table 19
Significance tests of the ERS-IWD and FPS-IWD algorithms using t-test with $\propto < 0.05$. Highlight (bold) denote that result is significantly different.

Data set	ERS-IWD	FPS-IWD	P-value
eil51	440.3	470.79	**5.17E−07**
eil76	551.8	617.91	**2.75E−10**
eil101	671.3	752.16	**1.30E−08**
ch130	6403.4	6820.03	**1.50E−08**
kroA100	22298.5	24162.10	**3.05E−10**
kroA200	32032.5	37316.04	**2.78E−13**
lin105	14801.8	15528.00	**7.84E−10**
lin318	47481.1	59324.57	**2.85E−19**

in Shah-Hosseini (2007). The results show that ERS-IWD outperformed FPS-IWD for all data sets, while LRS-IWD exhibited the worst results.

Tables 17–19 present the hypothesis test results, i.e., the t-test where $\propto < 0.05$. Table 17 shows that the average tour lengths of the 10 runs produced by ERS-IWD are significantly better than those from LRS-IWD. Table 18 compares the results between LRS-IWD and FPS-IWD. FPS-IWD significantly outperformed LRS-IWD for all eight TSPLIB data sets. Table 19 compares the results produced by ERS-IWD and FPS-IWD. ERS-IWD significantly outperformed FPS-IWD for all eight TSPLIB data sets. In general, it can be seen that the increase in complexity (the number of cities) of the TSPLIB data set results in increasing the significant difference of the performances.

To compare the computation time of the proposed methods, the average computation time of the lin105 data set (a large data set with 105 cities) was considered as an example. The average computation times of FPS-IWD, ERS-IWD, and LRS-IWD were 1.892, 26.071, and 6.269 min, respectively. Again, these results indicated that ranking-based methods were computational expensive. ERS-IWD appeared to be highly expensive in terms of computational time, which consumed more than 13 times as compared with that of the original IWD algorithm (FPS-IWD).

6. Conclusions

Selection methods have a significant impact on the performance of the IWD algorithm in undertaking optimization problems. In this paper, two ranking-based selection methods have been proposed for the IWD algorithm. The current IWD algorithm utilizes the FPS method in the solution construction phase. FPS is susceptible to a number of limitations (as explained in Section 1),

i.e. inability to accommodate negative soil values, sensitivity to the scaling function, selection domination, and inability to handle indistinguishable fitness. In this study, we replace the FPS method with two ranking-based selection methods i.e. linear ranking and exponential ranking, which are abbreviated as LRS-IWD and ERS-IWD, respectively. The main contribution of this research is to overcome the FPS limitations in preserving the search diversity of the IWD algorithm. The implication is that the proposed ranking-based selection methods help the IWD algorithm to avoid from being trapped in local optima during its search phase.

The proposed methods have been evaluated comprehensively using three combinatorial optimization problems, i.e. RSFS, MKP, and TSP. A systematic analysis has been conducted to determine the optimal settings of the most important parameter in LRS-IWD and ERS-IWD, i.e. the selective pressure (*SP*) that controls the gradient of the mapping function used in the selection methods. In general, the results indicate that the ranking-based selection methods have a direct impact on the efficiency of the IWD algorithm. This is important in practice when the IWD algorithm is used to solve complex real-world optimization problems. Based on the three optimization problems, ERS-IWD performs more effectively than LRS-IWD as well as the current FPS method. Nevertheless, ERS-IWD and LRS-IWD are computationally more expensive, as compared with the FPS method. In particular, the computational requirement of ERS-IWD is high (from 1.15 to 13 times higher than those of FPS-IWD as shown in the case studies), indicating the limitation of the exponential ranking-based selection method.

For future work, efficient techniques to reduce the computational complexity of ERS-IWD will be investigated. More experiments on other optimization problems need to be conducted too. Besides that, investigations pertaining to other aspects of optimization, such as a balance between exploration and exportation, are required. Hybridization of the IWD algorithm with other search techniques can also be examined. The applicability of the IWD algorithm to multi-objective optimization problems constitutes another direction of future work.

Acknowledgments

The first author acknowledges the Islamic Development Bank (IDB) for the granted scholarship to complete the Ph.D at the Universiti Sains Malaysia (USM). The authors would like to thank the anonymous referees for their careful reading of the paper and their valuable comments and suggestions, which greatly improved the paper.

References

Abdullah, S., & Jaddi, N. S. (2010). Great deluge algorithm for rough set attribute reduction. In *Database theory and application, bio-science and bio-technology* (pp. 189–197). Berlin: Springer.

Ahmed, H., & Glasgow, J. (2012). Swarm intelligence: Concepts, models and applications. 2012, Technical Report, 2012-585. Queen's University, Kingston, Ontario, Canada K7L3N6.

Akbarzadeh Totonchi, M. R. (2010). Intelligent water drops a new optimization algorithm for solving the vehicle routing problem. In *International conference on electronics and information engineering – ICEIE 2010*. Turkey.

Al-Betar, M. A. et al. (2012). Novel selection schemes for harmony search. *Applied Mathematics and Computation, 218*(10), 6095–6117.

Al-Betar, M. A. et al. (2013). An analysis of selection methods in memory consideration for harmony search. *Applied Mathematics and Computation, 219*(22), 10753–10767.

Alijla, B. O. et al. (2013). Intelligent water drops algorithm for rough set feature selection. In *Intelligent information and database systems* (pp. 356–365). Springer.

Azad, M. A. K., Rocha, A. M. A. C., & Fernandes, E. M. G. P. (2014). Improved binary artificial fish swarm algorithm for the 0–1 multidimensional knapsack problems. *Swarm Evolutionary Computation, 14*, 66–75.

Bae, C. et al. (2010). Feature selection with intelligent dynamic swarm and rough set. *Expert Systems with Applications, 37*(10), 7026–7032.

Bazan, J. G. et al. (2000). In *Rough set algorithms in classification problem, in rough set methods and applications* (pp. 49–88). Springer.

Bache, K., & Lichman, M. (2013). *UCI machine learning repository*. Irvine, CA: University of California, School of Information and Computer Science. Available from http://archive.ics.uci.edu/ml.

Chu, P. C., & Beasley, J. E. (1998). A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics, 4*(1), 63–86.

Colorni, A. et al. (1994). Ant system for job-shop scheduling. *Belgian Journal of Operations Research, Statistics and Computer Science, 34*(1), 39–53.

Dash, M., & Liu, H. (1997). Feature selection for classification. *Intelligent Data Analysis, 1*(1–4), 131–156.

Dorigo, M., & Di Caro, G. (1999). Ant colony optimization: A new meta-heuristic. In *Proceedings of the 1999 congress on evolutionary computation, CEC 99*. IEEE.

Duan, H., Liu, S., & Lei, X. (2008). Air robot path planning based on intelligent water drops optimization. In *IEEE international joint conference on neural networks, IJCNN 2008 (IEEE world congress on computational intelligence)*. IEEE.

Goldberg, D. E., & Deb, K. (1991). A comparative analysis of selection schemes used in genetic algorithms. *Urbana, 51*. 61801-2996.

Hendrawan, Y., & Murase, H. (2011). Neural-intelligent water drops algorithm to select relevant textural features for developing precision irrigation system using machine vision. *Computers and Electronics in Agriculture, 77*(2), 214–228.

Jensen, R., & Shen, Q. (2001). Rough and fuzzy sets for dimensionality reduction. In *Proceedings of the 2001 UK workshop on computational intelligence*.

Jensen, R., & Shen, Q. (2003). Finding rough set reducts with ant colony optimization. In *Proceedings of the 2003 UK workshop on computational intelligence*.

Jensen, R., & Shen, O. (2008). *Computational intelligence and feature selection: Rough and fuzzy approaches*. Wiley-IEEE Press. Vol. 8.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. Techn. Rep. TR06, Erciyes Univ. Press, Erciyes.

Lin, M.-H., Tsai, J.-F., & Yu, C.-S. (2012). A review of deterministic optimization methods in engineering and management. *Mathematical Problems in Engineering, 2012*, 15.

Machado, J. M. et al. (2001). A common tabu search algorithm for the global optimization of engineering problems. *Computer Methods in Applied Mechanics and Engineering, 190*(26), 3501–3510.

Mafarja, M., & Abdullah, S. (2011). Modified great deluge for attribute reduction in rough set theory. In *Eighth international conference on fuzzy systems and knowledge discovery (FSKD)*. IEEE.

Marler, R. T., & Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization, 26*(6), 369–395.

Nagalakshmi, P. et al. (2011). Combined economic and emission dispatch using intelligent water drops-continuous optimization algorithm. In *International conference on recent advancements in electrical, electronics and control engineering (ICONRAEeCE)*. IEEE: Sivakasi.

Niu, S. H., Ong, S. K., & Nee, A. Y. C. (2012). An improved intelligent water drops algorithm for achieving optimal job-shop scheduling solutions. *International Journal of Production Research, 50*(15), 4192–4205.

Passino, K. M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems, 22*(3), 52–67.

Pawlak, Z. (1982). Rough sets. *International Journal of Parallel Programming, 11*(5), 341–356.

Pawlak, Z. (1991). *Rough sets: Theoretical aspects of reasoning about data. System theory, knowledge engineering and problem solving* (Vol. 9). .

Reinelt, G. (1991). TSPLIB: A traveling salesman problem library. *ORSA Journal on Computing, 3*(4), 376–384.

Shah-Hosseini, H. (2007). Problem solving by intelligent water drops. In *IEEE congress on evolutionary computation, CEC 2007*.

Shah-Hosseini, H. (2008). Intelligent water drops algorithm: A new optimization method for solving the multiple knapsack problem. *International Journal of Intelligent Computing and Cybernetics, 1*(2), 193–212.

Shah-Hosseini, H. (2009). *Optimization with the nature-inspired intelligent water drops algorithm*. InTech, pp. 297–320.

Shah-Hosseini, H. (2012a). Intelligent water drops algorithm for automatic multilevel thresholding of grey-level images using a modified Otsu's criterion. *International Journal of Modelling, Identification and Control, 15*(4), 241–249.

Shah-Hosseini, H. (2012b). An approach to continuous optimization by the intelligent water drops algorithm. *Procedia-Social and Behavioral Sciences, 32*, 224–229.

Shi, Y. (2001). Particle swarm optimization: Developments, applications and resources. In *Proceedings of the 2001 congress on evolutionary computation*. IEEE.

Srinivas, M., & Patnaik, L. M. (1994). Genetic algorithms: A survey. *Computer, 27*(6), 17–26.

Vafaie, H., & Imam, I. F. (1994). Feature selection methods: Genetic algorithms vs. greedy-like search. In *Proceedings of international conference on fuzzy and intelligent control systems*.

Vesterstrom, J., & Thomsen, R. (2004). A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In *Congress on evolutionary computation, CEC2004*. IEEE.

Wang, X. et al. (2007). Feature selection based on rough sets and particle swarm optimization. *Pattern Recognition Letters, 28*(4), 459–471.

Weise, T. et al. (2009). *Why is optimization difficult?, in nature-inspired algorithms for optimisation*. Springer.

Wong, L.-P., Low, M. Y. H., & Chong, C. S. (2010a). Bee colony optimization with local search for traveling salesman problem. *International Journal on Artificial Intelligence Tools, 19*(03), 305–334.

Wong, L.-P., Low, M. Y. H., & Chong, C. S. (2010b). A generic bee colony optimization framework for combinatorial optimization problems. In *Fourth Asia international conference on mathematical/analytical modelling and computer simulation (AMS)*. Kota Kinabalu, Malaysia: IEEE.

Yang, X. S. (2008). *Nature-inspired metaheuristic algorithms*. Luniver Press.

Yildiz, A. R. (2009). An effective hybrid immune-hill climbing optimization approach for solving design and manufacturing optimization problems in industry. *Journal of Materials Processing Technology, 209*(6), 2773–2780.

Yu, L., & Liu, H. (2004). Efficient feature selection via analysis of relevance and redundancy. *The Journal of Machine Learning Research, 5*, 1205–1224.

Zaiyadi, M. F., & Baharudin, B. (2010). A proposed hybrid approach for feature selection in text document categorization. *World Academy of Science, Engineering and Technology, 48*, 111–116.