https://doi.org/10.1080/10170660709509062

WATER FLOW-LIKE ALGORITHM FOR OBJECT GROUPING PROBLEMS

Feng-Cheng Yang* and Yuan-Peng Wang
Graduate Institute of Industrial Engineering
National Taiwan University
1, Sec. 4, Roosevelt Road, Taipei, 106 Taiwan

ABSTRACT

This paper presents a novice heuristic algorithm, Water Flow-like Algorithm (WFA), for solving discrete optimization problems, particularly the bin packing problems. WFA simulates solution agents as water flows traversing the terrain mapped from the objective function. Governed by the gravitation force, water flows from higher attitudes to lower ones. Driven by the fluid momentum, water flows adjust their compositions and directions against the landforms by splitting into and merging from other flows. Water flows are allowed to move upward to higher attitudes once they possess enough momentum to overcome the potential barrier. Mostly, at least one flow can travel to the lowest region of the terrain under the consideration. In the atmosphere, some water of a flow will evaporate and return to the ground by precipitation. Inspired by the water flowing of the nature, WFA is designed as an optimization algorithm performing the water flow splitting, merging, and dropping (precipitation) operations to traverse the solution space. The number of solution agents deployed is dynamically changing. WFA is an evolutionary algorithm involving four water flow operations: splitting and moving, merging, evaporation, and precipitation. The computational flow and the four operations are extensively discussed. In addition to general operations of WFA, specific operations for bin packing problems are presented. A designed problem and a benchmark problem from OR-Lib are used to test WFA and to compare results with other methods, such as GA, POS, and EM. Numerical results show that WFA outperforms others in solving these BPPs.

Keywords: heuristic algorithm, discrete optimization, water flow-like algorithm, flow merging, flow splitting

1. INTRODUCTION

Heuristic optimization methods have been widely used to solve complicated problems for a couple of decades. The problems targeted are normally real-world problems involving nonlinear mathematics or integer-and-real-mixed programming models. Most of the developed heuristic methods conduct solution search operations to systematically explore the solution space. For general optimization algorithms, such as Generic Algorithm and Ant Colony Optimization methods, stochastic approach is usually adopted in the searching operation to cover the solution space as complete as possible. In addition, agent technology is also extensively used in these algorithms. Single and multiple agent techniques can be found in many algorithms. The well-known simulated annealing algorithm is a typical single-agent method, while the GA is a multiple one. Nowadays, multiple-agent-based methods are widely used to fully utilize the computation power of current techniques. Group intelligence emerges from the population of agents facilitating the search for the near optimum efficiently.

In general, a single-agent-based heuristic algorithm conservatively searches the solution space step by step. Although the search might be inefficient, gradients of the objective function can be obtained and used to effectively guide the search direction toward prominent areas. Therefore, the method is suitable for optimization problems with a smooth solution space; but not for complex problems with multiple local optima. On the other hand, although multiple-agent-based algorithms can efficiently search the solution space using the group intelligence, they suffer the quick convergence and redundant searches. Resource is wasted in unavoidable redundant searches, especially for those methods having several heuristic models mapping to a single solution. Neither the single nor the multiple agent method is agile enough to conduct an efficient and effective solution search. This paper proposes an improving idea of using a size-not-fixed population of agents to overcome this

* Corresponding author: iefcyang@ntu.edu.tw

deficiency. Therefore, the number of agents deployed for solution searching is deterministically changing. The local optimality of the solution space determines whether the method should conduct agent forking or merging operation.

The design of the proposed optimization method is inspired by the natural behaviors of water flowing from higher levels to lower ones. A flow will spit into multiple subflows when rugged and bumpy terrains are traversed. Conversely, subflows merge with each other when they join at the same location. Governed by the gravitation force and driven by the fluid momentum, flows are able to run up to higher levels or run over bumps to traverse various areas of the terrain. Water flowing will cease and stagnate at locally or globally lowest depressions, when the momentum left cannot expel the water out off the depressions. Once the solution space of a minimization problem is mapped as the geographical terrain and the objective value is mapped to the attitude, then each flow can be regarded as a solution agent. The solution search is thus modeled as water flowing. Since the number of flows is changing, this method is intrinsically an agent population varying method.

The contents of this paper are arranged as follows. Section 2 briefs the operations of several well-known heuristics algorithms. The proposed WFA is presented in Section 3. Experimental tests for solving bin packing problems using WFA are given in Section 4, in which results are also compared with other methods. Conclusions are presented and suggestions for future research are given in the last section.

2. LITERATURE REVIEWS

Most of the current heuristic optimization methods are usually adapted from or inspired by natural behaviors or phenomena. These behaviors or phenomena are largely succeeded from the biology, zoology, and physics. A few popular methods are introduced as follows.

Tabu search: This optimal solution search method was first presented by Glover in 1977 and then populated with detailed computation algorithm subsequently [5, 6]. The method can be regarded as a single agent approach with a limited intelligence of knowing the searched solutions, which are kept in a tabu list. This method had been widely applied in those days that the computation power was not yet boosted by the current technology. Like many heuristic methods, tabu search also suffers from the pitfalls of being stuck in local optima [16].

Genetic Algorithm, GA: GA, proposed by Holland [8], is the most popular and widely used heuristic method for solving various kinds of optimization problems [4]. The solution search method mimics the biological evolution, where the inferior solutions are replaced with superior ones generation by generation. Crossover and mutation operations of GA are adopted and adapted from the genetics to generate (search for) new solutions. Selection operation is employed to evolve the solutions to ones with higher fitness. Much research has successfully used GA to solve various optimization problems. In particular, Falkenauer [3] used a hybrid GA to solve bin packing problems of various degrees of complexity. Results showed the GA was suitable for solving BPPs of medium sizes.

Particle Swarm Optimization, PSO: PSO was first presented by Kennedy and Eberhart [9, 10]. The design of PSO was inspired by the group intelligence resulting from the behaviors of a flock of animals, such as fishing behaviors of birds. Through communication between individuals and global information sharing, the population of animals is able to achieve their goal. PSO was originally designed to solve continuous optimization problems, either single- or multi-objective problems [11]. Kennedy [9] has proposed a clustering analysis method to enhance the efficiency of PSO. Shi and Eberhart [15] redefined the position and velocity of particles to include fuzzy concept to enhance the computation efficiency.

Electromagnetic-like mechanism, EM: EM optimization algorithm is a new heuristic optimization method, which was developed by Birbil and Fang [1]. This algorithm adopts the agent collaboration and evolution techniques to search for the global optimums. The agents deployed in this method are “particles” that carry electrical charges and exert electromagnetism forces from other particles. In addition, the solution space is regarded as a hyperspace of electromagnetism field. The positions of the particles are solutions to the problem. An artificial electromagnetism field is built to propel the particles to new positions. Within this field, magnetism forces of attraction and repulsion from other particles are exerted on each particle to move it to a “better” position. This procedure is then recursively executed until an acceptable solution is obtained. EM is utilized to solve continuous problems and the effectiveness has been shown promising. General EM-based discrete optimization techniques were developed by Yang, Kang et al. [13]. The application of EM to solve object sequencing and grouping problems were also presented and discussed in [14].

In addition to the previous heuristic methods, several popular heuristic methods are available for solving general and specific optimization problems, such as simulated annealing, branch-and-bound, ant colony optimization. Since they are not referenced in this paper, they are not covered here.

Each heuristic method has distinct merits on its ad hoc designs of algorithmic operations. Nevertheless, each also suffers various deficiencies when facing peculiar, large-size, and complicated problems. Currently, agent techniques are widely used in heuristic methods. In single-agent-based methods, only one solution agent is dispatched, such as simulated annealing and tabu search. Although, the computation cost is light yet the solution search is inefficiency. The wideness of the solution search is restricted. For multiple-agent-based methods, such as GA, PSO, and EM, a size-fixed population of agents is dispatched for efficient solution search. However determining the proper population size is not easy. Computation resource is easily wasted in redundant searches when a large number of agents are deployed. Current heuristic methods therefore share two common difficulties in the optimum solution search. (1) Convergence to local optima: Since the evolution tension is intrinsically high, inferior solutions are easily discarded and the solution agents flock together at a local optimum. (2) Jumpy searches without further exploitation on the solution space: Almost all of the methods are able to perform lively solution search but lack of fine grained exploitation to obtain better solutions. The easier method is to conduct local searches, however it is costly.

This paper proposed a new heuristic method that employs a size-not-fixed population of agents to steadily search optimum solutions and to balance the exploration and exploitation of the solution search.

3. WATER FLOW-LIKE ALGORITHM

This paper presents a water-flowing inspired optimization algorithm, *water flow-like algorithm* (WFA for short). Solution agents are modeled as water flows and the objective function surface on the solution space is modeled as the terrain traversed by the flows. Behaviors of fluid flows have inspired the design of WFA, specifically in the design of solution search. The movements of the flows (location changes) are governed by the gravitation force and the energy conservation law. When the solution search starts, only one water flow is deployed; i.e., a single agent is assigned. Afterwards, the flow splits into multiple subflows when rugged or broken terrains are encountered. Conversely, a number of flows will merge into one flow when they join together at the same location. Therefore, the number of flows (agents) is changing in quest for the optimal solutions.

(1) Adopted Behaviors of Fluid Flows in WFA

When water flows are regarded as solution agents and the landform is mapped into the surface of the objective function, a water moves to a lower position can be considered as a solution search for optima. Several natural behaviors of water flows [2] have been adopted in designing WFA. These behaviors are presented as follows with a brief discussion of the design ideas of WFA.

1. Driven by the gravitation force and governed by the energy conservation law, water will constantly flow to lower attitudes. Therefore, the solution search will recursively move to superior solutions from inferior ones iteration by iteration.
2. Fluid momentum drives water moving forward against rough terrains. A flow will split into subflows when it encounters rugged terrains and its momentum exceeds an amount for splitting. WFA simulates this behavior as an agent forking operation; i.e., more than two agents are derived from a single agent. A flow with larger momentum will generate more streams of subflows than a lower one. A flow with limited momentum will yield to the landform and maintain a single flow. Therefore, the fluid momentum of a flow is recalculated for determining the number of subflows that can be forked after each move.
3. Water flows to lower attitudes and occasionally springs up or swells up to higher attitudes, once the kinetic energy is larger than the required potential energy. To avoid being trapped within a local minimum, WFA allows the water to flow to a worse location to broaden the exploration area.
4. A number of flows merge into a single flow when they run to the same location. WFA reduces the number of solution agents when multiple agents move to the same location, to avoid redundant searches.
5. Water flows are subject to water evaporation in the atmosphere. The evaporated water will return to the ground by rainfalls. In WFA, a part of the water flow is manually removed to mimic the water evaporation. A precipitation operation is implemented in WFA to simulate natural rainfall to explore a wider area.

The discussed properties and behaviors of water flows have been adopted in WFA. Notice that, an agent number changing strategy is used in this heuristic algorithm, which is also a novice approach in heuristic optimization technology.

(2) Operations of WFA

<u>Overview of WFA</u>

In the beginning, only one water flow is defined at a randomly generated location. In addition, the initial mass and velocity are given. The location of the initial flow is then the initial solution to the optimization problem. The value of objective function computed from a location is regarded as the attitude of the location. Driven by the fluid momentum and the potential energy, water-flows flow to new locations to traverse the solution space for new solutions.

In this paper, the movement of a flow is designed as a constant-step movement to the best neighboring location for solving object grouping problems. Various flow moving strategies can be applied for different optimization problems. Velocity of a flow is not used to determine the new location of the flow. It is, however, only used to identify whether the flow can move. Flow splitting is resulted from a capable momentum. A flow with a higher momentum generates more subflows than a lower one. The locations of the split subflows are derived from the neighboring locations of the original flow. A tabu list of recently searched locations is checked to discard infeasible and traversed locations from a set of neighboring locations. The available neighboring locations are evaluated and ranked. If a flow will split into *n* subflows, the first *n* ranked locations are assigned for these subflows. When no splitting happens to a flow, the flow flows as a single stream to the best feasible neighboring location. In contrast, if a flow has a zero velocity, it will stagnate at its location. No movement is allowed until other flows merge it with velocity or the water of the flow evaporates to the atmosphere and returns later back to the ground.

The energy conservation law is followed in WFA to assign the mass and velocity to each subflow in flow splitting. The mass of the original flow is divided and assigned to each subflow, based on its rank. The velocity of each subflow is calculated referring to the attitude drop or rise from the original flow to the subflow. In each iteration, when subflows are split from the original flow they will behave as individual flow and the original one is discarded.

In general, most of the heuristic algorithms might initiate a *reset* or *mutate* operation to explore more solution space or escape from traps of local optimums. To mimic the natural behaviors and to widely explore the solution space, evaporation and precipitation operations are included in WFA. Water in the water flows will constantly evaporate into the atmosphere. When the evaporated water accumulates to some extend, it will return to the ground as several new pour-downed flows. Therefore rainfalls will occur occasionally.

As discussed, solution agents are modeled as water flows in WFA and the number deployed is dynamically changing. Figure 1 shows the computation flow of WFA. Notice that, (1) flow splitting and moving, (2) flow merging, (3) water evaporation, and (4) precipitation are the four primary operations in WFA.

***Flow splitting and moving*:** Driven by the gravitation force and govern by the energy conservation law, water flows flow to new locations. In nature, water cascades to multiple streams when it flows from higher attitudes to lower ones in a rugged terrain. In WFA, a flow is regarded as a solution agent and it can fork multiple agents to succeed the solution search. A flow movement is actually a solution search from the current location to a new one. In WFA, the design of flow moving operation is problem-dependent. Different problem types will have their own designs in the flow moving operations. In particularly, this paper presents one for the object grouping problems.

***Flow merging*:** When a number of flows flow to the same location, they will join together as a single flow to continue the journey for searching optimal solutions. Occasionally, the merging operation will merge stagnant flows with live flows to remove the stuck or trapped flows.

***Water evaporation*:** A small portion of the water of a flow will be continuously removed to simulate water evaporation to the air. The amount of evaporated water keeps cumulating until a rainfall condition is met. The operation is to prepare for flow regenerations to increase the wideness of solution search.

***Precipitation*:** This operation is executed periodically to simulate rainfalls in the nature. A new population of flows is generated and added to the existing one to explore more areas of the solution space. Occasionally, precipitation is conducted to bring livingness back to the current flows when all of them are stagnant.

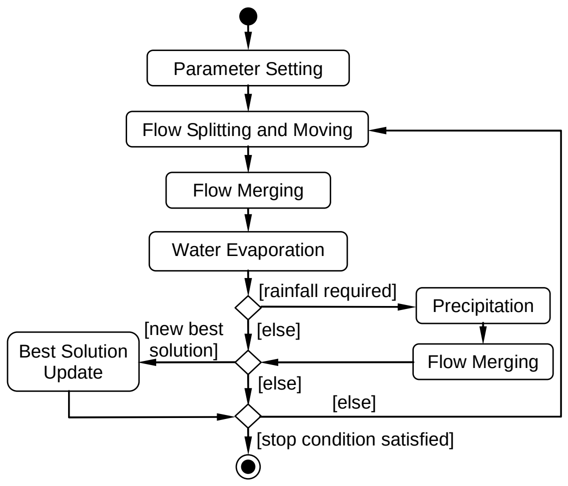


Figure 1. Computation flow of WFA

The above four operations are iteratively and conditionally executed in each computation iteration until an acceptable solution is obtained. Detailed computing procedures will be discussed after presenting the required parameters for WFA.

Parameters Defined and Used in WFA

Parameters required for WFA execution includes: the iteration limit, *P*; the initial mass, M_0, and velocity, V_0, of the original flow; the base momentum, T; the limit of the number of subflows split from a flow, $\bar{n}$; the upper bound, $b_h^{(u)}$, lower bound, $b_h^{(l)}$, neighboring step size, r_h, ant precipitation offset, e_h, of each coordinate h; and the number of iterations, t, for periodical precipitation. The default stopping condition for WFA is the reach of iteration limit *P*. The number of forked subflows from a flow is determined from the value of dividing its momentum by the base moment T. The initial mass and velocity is set by the user; yet M_0V_0 =2T ~3T is suggested. The number of subflows split from a flow is subject to an upper limit $\bar{n}$; and $\bar{n}=3$ is suggested to avoid a skyrocket increase of the number.

Flow moving operation in WFA is dependent on the targeted problems. This paper presents a *neighboring advancement* method for object grouping problems. The constant step size r_h is specified to locate the neighboring locations. Smaller step size will yield a fine-grained solution search, yet an inefficient operation. Since no granularity issue is involved in discrete optimization problems, $r_h = 1$ is typically specified to cover all possible neighboring solutions. Since a periodical precipitation operation for t iterations is executed, to avoid overwhelmed flow regenerations, $t > P/20$ is suggested. In addition, the coordinate offset for assigning coordinate h of the location for pour-downed flows in precipitation is suggested to be $e_h = b_h^{(u)} - b_h^{(l)}$, to cover the entire solution space. The following subsections discuss the four operations of WFA in details: flow splitting and moving, flow merging, water evaporation, and precipitation.

Flow Splitting and Moving

The primary feature of WFA is the forking of solution agents, which is executed by the operation of water splitting. When water flows down to lower attitudes, the gained potential energy will convert to kinetic energy increasing the velocity and the chance for flow splitting. The flow with a higher kinetic energy will split more subflows to traverse more area than a lower one.

Let *N* be the number of water flows in the current iteration. In the water flow operation, the number of subflows n_i forked from flow i ($i \in \{1, 2, ..., N\}$) is determined by its momentum, M_iV_i. No water splitting happens to a flow with zero momentum. Nevertheless, a flow can split into subflows only when its momentum exceeds a certain limit. The basic momentum *T* is thus defined for computing the number of subflows that a flow can split. For example, if M_iV_i=3.45*T*, the number of subflows is 3. Therefore, if $0 < M_iV_i < T$, flow i will solitarily move to a new location without splitting. If the number of subflows is not restricted in flow splitting, the computation resource will be used up. Therefore, an upper limit $\bar{n}$ of the number of subflows forked from a flow is imposed. To fulfill this design, the number of subflow split from flow i is

$$n_i = \min\left\{\max\left\{1, \operatorname{int}\left(\frac{M_iV_i}{T}\right)\right\}, \bar{n}\right\}. \tag{1}$$

Since the presented WFA is mainly designed for solving discrete optimization problems, the flow movements to new locations are not calculated from Euclidean distances. Instead, neighboring locations are allocated to move the flow or for flow splitting. Suppose that flow i represents solution X_i and the current solution set is $\mathbf{X}$, $\mathbf{X} = \{\mathrm{X}_1, \mathrm{X}_2, ..., \mathrm{X}_N\}$. Note that the size N and contents of $\mathbf{X}$ will change after the splitting and moving operation.

A neighboring location is computed from the current location of the flow by forwarding or backwarding a constant step on a coordinate. Suppose that X_i is represented as a position vector, $\mathrm{X}_i = \left[x_{i1}\, x_{i2} \cdots x_{iq}\right]$, where q is the number of variable, coordinate $x_{ih} \in \left[b_h^{(l)}, b_h^{(u)}\right]$, and r_h is the step size of coordinate h and solution. Then,

$$\mathrm{A}_{ih}^{(+)} = \left[x_{i1}\, x_{i2} \ldots \min\left\{x_{ih} + r_h, b_h^{(u)}\right\} \ldots x_{iq}\right] \tag{2}$$

and

$$\mathrm{A}_{ih}^{(-)} = \left[x_{i1}\, x_{i2} \ldots \max\left\{x_{ih} - r_h, b_h^{(l)}\right\} \ldots x_{iq}\right] \tag{3}$$

are two neighboring solutions generated from one step moving of coordinate h. Let sets

$$\mathbf{A}_i^{(+)} = \left\{\mathrm{A}_{i1}^{(+)}, \mathrm{A}_{i2}^{(+)}, ..., \mathrm{A}_{iq}^{(+)}\right\} \tag{4}$$

and

$$\mathbf{A}_i^{(-)} = \left\{\mathrm{A}_{i1}^{(-)}, \mathrm{A}_{i2}^{(-)}, ..., \mathrm{A}_{iq}^{(-)}\right\}, \tag{5}$$

the set of all one-step neighboring solutions for flow i is

$$\mathbf{A}_i = \mathbf{A}_i^{(+)} \cup \mathbf{A}_i^{(-)}. \tag{6}$$

Therefore, $\mathbf{A}_i$ has at most $2q$ neighboring solutions for X_i. The solutions (locations) in $\mathbf{A}_i$ might have been traversed by flow i. To avoid redundant solution searches, a tabu list $\mathbf{H}_i$ of recently searched locations is updated for flow i, where $\mathbf{H}_i = \{ \mathrm{H}_i^{(1)}, \mathrm{H}_i^{(2)}, \mathrm{H}_i^{(3)} \}$. Note that $\mathrm{H}_i^{(1)}$, $\mathrm{H}_i^{(2)}$, and $\mathrm{H}_i^{(3)}$ are the locations traversed in one, two, and threes iterations before moving to the current location X_i. Removing the traversed location yields the candidate set of neighboring solution $\mathbf{C}_i$ for flow i, $\mathbf{C}_i = \mathbf{A}_i - \mathbf{H}_i$. The objective function values of the solutions in $\mathbf{C}_i$ are then subsequently evaluated and sorted from superior solutions to inferior ones. The best n_i solutions in $\mathbf{C}_i$ compose the set of subflows for flow i as $\mathbf{U}_i = \{ \mathrm{U}_{i1}, \mathrm{U}_{i1}, ..., \mathrm{U}_{in_i} \}$, where U_{ik} is the location of the kth best subflow split from flow i. The tabu list of flow i is updated with the current location and then delivered to each subflow: $\mathrm{H}_i^{(3)} \leftarrow \mathrm{H}_i^{(2)}$, $\mathrm{H}_i^{(2)} \leftarrow \mathrm{H}_i^{(1)}$, and $\mathrm{H}_i^{(1)} \leftarrow \mathrm{X}_i$. As mentioned previously, energy conservation is considered in splitting flow i into n_i subflows. The mass of flow i is discriminately distributed into subflows based on their ranks. The mass distributed from M_i to the subflow U_{ik} is

$$w_{ik} = \left(\frac{n_i + 1 - k}{\sum_{r=1}^{n_i} r} \right) M_i, \; k = 1, 2, ..., n_i. \quad (7)$$

For example, if $M_i = 5$ and $n_i = 3$,

$w_{i1} = \left(\frac{3}{1+2+3} \right) 5$, $w_{i2} = \left(\frac{2}{1+2+3} \right) 5$, and $w_{i3} = \left(\frac{1}{1+2+3} \right) 5$. The velocity of each subflow is then computed from equation of energy conservation. Velocity of subflow U_{ik} split from flow i is

$$\mu_{ik} = \begin{cases} \sqrt{V_i^2 + 2g\delta_{ik}}, & if \quad V_i^2 + 2g\delta_{ik} > 0 \\ 0, & otherwise \end{cases}, \quad (8)$$

where g is the gravitational acceleration,

$$\delta_{ik} = \begin{cases} f(\mathrm{X}_i) - f(\mathrm{U}_{ik}), \text{for minimization} \\ f(\mathrm{U}_{ik}) - f(\mathrm{X}_i), \text{for maximization} \end{cases}, \quad (9)$$

and $f(\mathrm{U}_{ik})$, $f(\mathrm{X}_i)$ are objective function values of subflow U_{ik} and flow i, respectively. Note that δ_{ik} is the attitude drop from X_i to U_{ik}. When $V_i^2 + 2g\delta_{ik} < 0$, the momentum delivered to subflow U_{ik} has been used up. This subflow will stagnate in its location, no splitting and no movement happens until other flows merge it and carry it away or the evaporation operation remove it to the air.

When subflows are generated from a flow, the original flow is discarded. Therefore, the solution set is updated by $\mathbf{X} \leftarrow \bigcup_{i=1}^{N} \mathbf{U}_i$ and the number of current flows is updated as $N \leftarrow \sum_{i=1}^{N} n_i$. The augmented set is then subject to a flow merging operation to merge those flows having the same location.

A special case may happen that all subflows receive zero velocities. Since no any flow movement can be conducted, they will remain on the ground and gradually evaporated into the atmosphere. Finally, the evaporated water will return to the ground by precipitation. To avoid unnecessary calculation, a forced precipitation operation is executed for this special case. Detailed procedures will be covered later.

Flow Merging

When more than two flows of water move to the same location, they will merge into a single flow. Masses and momentums are therefore accumulated to compose an aggregated flow to reinforce the solution search around the location. Flow merging may bring a stagnant flow live energy to escape from the trapped location. This operation systematically checks a flow with others whether they share the same location. If it does, the latter flow is merged into the former one. Suppose that flows i and j share the same location (i.e., $\mathrm{X}_i = \mathrm{X}_j$), flow j is merged into flow i and thus flow j is discarded. The mass of flow j is added to flow i,

$$M_i \leftarrow M_i + M_j \quad (10)$$

and aggregated velocity for flow i is

$$V_i \leftarrow \frac{M_i V_i + M_j V_j}{M_i + M_j}. \quad (11)$$

As noted, the computation of the aggregated velocity follows the law of momentum conservation. The tabu list of flow i is then updated by assigning $\mathrm{H}_i^{(2)}$ as $\mathrm{H}_i^{(3)}$ and replacing $\mathrm{H}_i^{(2)}$ with $\mathrm{H}_k^{(1)}$; i.e., $\mathrm{H}_i^{(3)} \leftarrow \mathrm{H}_i^{(2)}$ and $\mathrm{H}_i^{(2)} \leftarrow \mathrm{H}_k^{(1)}$. The flow merging operation is thus executed right after an augmented solution set $\mathbf{X}$ is resulted to eliminate redundant flows.

Water Evaporation

To simulate the natural evaporation of water into the air, a water evaporation operation is executed in WFA. Each water flow is subject to water evaporation, where a portion of the water is evaporated into the atmosphere. After a prescribed number of iterations, the evaporated water will return to the ground by precipitation. The returned locations are stochastically determined from the locations of the current flows. In water evaporation operation the masses of all flows are updated by

$$M_i \leftarrow \left(1-\frac{1}{t}\right)\bar{M}_i, i=1,2,...,N\ , \tag{12}$$

where $\bar{M}_i$ is the mass when flow i is initially generated or merged with others. Therefore, the total mass of a flow will be completely removed and thus discarded after t iterations, when the flow is not subject to merging or splitting.

Precipitation

Two types of precipitation are executed in WFA to simulate the natural life cycle of water. The first is an *enforced precipitation* used in propelling energy to grounded flows (stagnated flows). The other is a *regular precipitation* which will be conducted once in t iterations.

Enforced Precipitation: This precipitation is applied when all flows are grounded with zero velocities. All flows are forced to completely evaporate into the atmosphere and then return to the ground without changing the number of current flows. However, the returned locations are deviated stochastically from the original ones. In addition, the mass of each flow is proportionally distributed and velocity is initialized with the initial velocity. Let the new location of flow i is $X'_i = \left[x'_{i1}\, x'_{i2} \cdots x'_{iq}\right]$. Each component x'_{ih} is obtained stochastically from the original coordinate x_{ih} by

$$x'_{ih} = \begin{cases} x_{ih} + d_h^+, & if \sim U(0,1) > 0.5 \\ x_{ih} - d_h^-, & otherwise \end{cases}, \tag{13}$$

Where

$$d_h^+ = \begin{cases} \varepsilon_h, & if\ b_h^{(u)} \geq \ \varepsilon_h + x_{ih} \\ 0, & otherwise \end{cases}, \tag{14}$$

$$d_h^- = \begin{cases} \varepsilon_h, & if\ b_h^{(l)} \leq \ x_{ih} - \varepsilon_h \\ 0, & otherwise \end{cases}, \tag{15}$$

$$\varepsilon_h = \sim U(0,1)\cdot e_h\ , \tag{16}$$

and e_h is the offset limit of coordinate h. In Eqs. 13 and 16, $\sim U(0,1)$ is a run-time generated normalized variate, which is uniformly distributed. Note that the coordinates of the new location are either remaining within the bounds or remaining the same as the original ones. Mass of M_0 is proportionally distributed to the flows based on their original mass. Consequently, the mass assigned to flow i is

$$M'_i = \left(\frac{M_i}{\sum_k M_k}\right) M_0\ ,\ i = 1,2,...,N\ . \tag{17}$$

In this case, since the operation has relocated all of the current flows, the precipitation operation is like a reset of computation. However, the reset is in fact regenerating multiple flows, rather than resetting to the initial condition. After the mass has be distributed, the initial velocity V_0 is assigned to each flow, $V'_i \leftarrow V_0$. Although the location is relocated, the tabu list of the three recently traversed locations is kept the same for each flow. Let $\mathbf{X}'$ be the new solution set, $\mathbf{X}' = \{X'_1, X'_2, ..., X'_N\}$; the set of current flows is then updated by $\mathbf{X} \leftarrow \mathbf{X}'$.

Regular Precipitation: Regular precipitation is applied periodically to move evaporated water back to the ground. In t iterations, the operation executes once to pour down the evaporated water. This operation can be regarded as a wide exploration on the solution space. In this operation new flows are generated and added to the current solution set. Various designs of this operation can be proposed. The tasks include defining the number of new flows and assigning locations for them. To elaborate the solution exploration, this paper pours down N new flows to the ground, which will double the number of the current flows. Let set $\bar{\mathbf{X}} = \{\bar{X}_1, \bar{X}_2, ..., \bar{X}_N\}$ be the set of flows poured down. Similar to the location derivation of the enforced precipitation, the location $\bar{X}_i$ of a pour-downed flow i is stochastically derived from the location of the ground flow i, the counter part, where

$$\bar{X}_i = \left[\bar{x}_{i1}\, \bar{x}_{i2} \ldots \bar{x}_{iq}\right], \tag{18}$$

$$\bar{x}_{ih} = \begin{cases} x_{ih} + d_h^+, & if \sim U(0,1) > 0.5 \\ x_{ih} - d_h^-, & otherwise \end{cases}. \tag{19}$$

Notice that the cumulated mass of the evaporated water is $M_0 - \sum_k M_k$. Referring to the masses of the ground flows, the masses of the pour-downed flows are proportionally assigned by

$$\bar{M}_i = \left(M_0 - \sum_k M_k\right)\frac{M_i}{\sum_k M_k} = \left(\frac{M_0}{\sum_k M_k} - 1\right)M_i, i = 1,2,...,N. \quad (20)$$

The velocity for each pour-downed flow is initialized with the initial velocity, $V_i' \leftarrow V_0$. The tabu list of the pour-downed flow i is copied from the existing ground flow i. At the end of the operation, the set of current flows is augmented by the union of the ground flows and pour-downed flows; i.e., $\mathbf{X} \leftarrow \mathbf{X} \cup \bar{\mathbf{X}}$ and $N \leftarrow 2N$.

Notice that both of the enforced and the regular precipitation operations might assign multiple flows to the same locations. Therefore, a flow merging operation is executed right after the precipitation operation to eliminate the redundant flows.

The solutions in $\mathbf{X}$ are evaluated to find the iteration-best solution $\tilde{\mathrm{X}}$ after the operations of precipitation. Result comparison between $\tilde{\mathrm{X}}$ and the best solution so far, X^*, is then conducted. Once the objective value $f(\tilde{\mathrm{X}})$ is better than $f(\mathrm{X}^*)$, the best solution so far is replaced with the iteration-best solution; i.e., $\mathrm{X}^* \leftarrow \tilde{\mathrm{X}}$.

As discussed in the previous subsections, the water moving operations are iteratively executed until the specified stop condition is satisfied. A simple condition is the reach of a specified number of iterations. An alternative is the reach of a specified number of objective function calls, which is used in this paper for comparisons with other methods. The best solution so far, X^*, is then occasionally updated during the computation until the end of the WFA execution.

(3) Design of WFA for the Object Grouping Problem

The object grouping problem discussed in this paper is the bin packing problem, BPP. Bin packing problems with a tight capacity constraint have challenged several heuristic methods for feasible and optimal solutions. The traditional bin packing problem is to minimize the number of bins used subject to a weight capacity constraint. The BPPs discussed in this paper, however, is to balance the weights between bins with or without weight capacity constraints.

Therefore, the objective is to minimize the weight deviation between bins and hope for a minimal number of overloaded bins when weight constraint is considered.

Most of the standard operations of WFA discussed above can be directly applied to solve BPPs. If the number of object to be packed is q, the solution to the BPP can be derived from the location of flow i, $\mathrm{X}_i = \left[x_{i1} x_{i2} \cdots x_{iq}\right]$. Notice that, coordinate x_{ih} is therefore the bin ID where object h is pack into. Let ξ be the number of bins used. Then $x_{ih} \in \left[b_h^{(l)}, b_h^{(u)}\right] = \left[1, \xi\right]$, for all $h = 1,2,...,q$. Since the neighboring advancement strategy of assigning neighboring solutions as the locations of the split subflows is used, the step size $r_h = 1$ is set. Moreover, $e_h = b_h^{(u)} - b_h^{(l)} = \xi - 1$ for allocating locations of the pour-downed flows. Operations of flow splitting, merging, water evaporation, and precipitation for BPPs are exactly the same as discussed above except that the values of the coordinates must be integers. Therefore, Eq. 16 should be replaced with

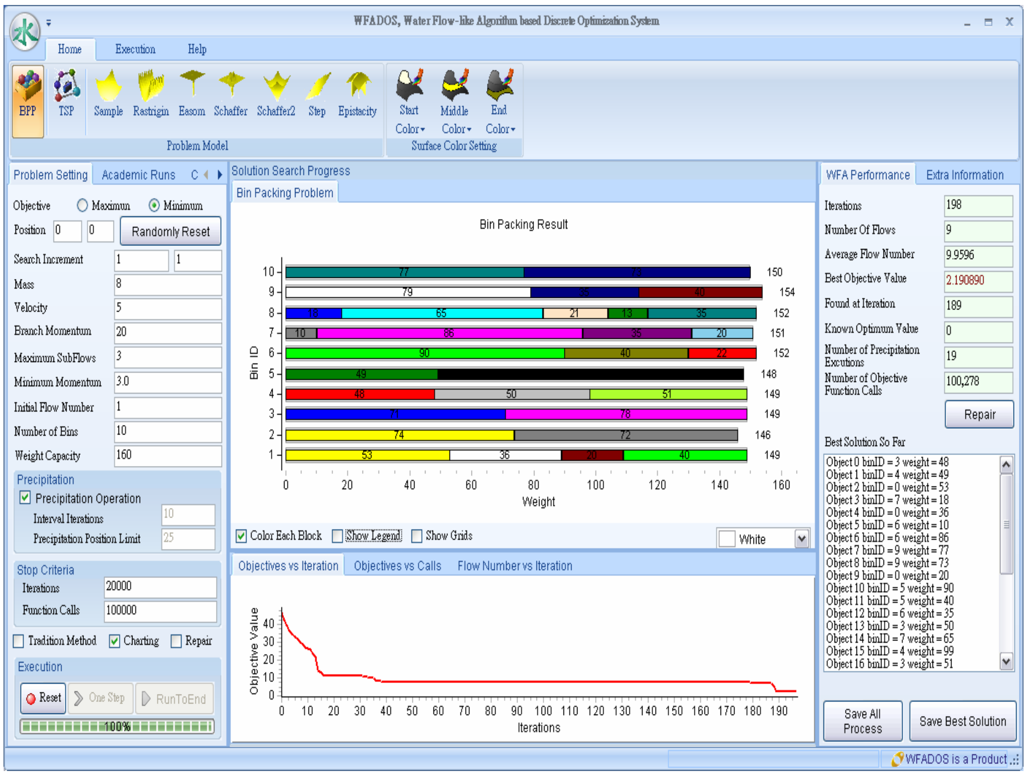
$$\varepsilon_h = \mathrm{int}\left(\sim U\left(0,1\right) \cdot e_h\right), \quad (21)$$

to guarantee the locations of the pour-downed flows are located at integral coordinates.

4. WFADOS AND NUMERICAL EXPERIMENTS

In addition to the proposed algorithm, this research has developed a WFA-based discrete optimization system, the WFADOS (Water Flow-like Algorithm based Discrete Optimization System). Figure 2 is a snapshot of the system.

Numerical examples of BPPs tested in this paper include a designed BPP and a benchmark from the OR-Library [12]. They were used to evaluate the effectiveness of WFA. The WFADOS was used to solve these problems and the results were compared with GA, PSO, and EM. To fairly compare the performances and the numerical results between WFA and other heuristic methods, the number of objective function calls was set to the same and served as the stopping criterion for all methods. To let all the heuristic methods unfold their own merits in the competition, a high limit of objective function calls was set to guarantee all methods had obtained their final best solutions. Therefore, several pilot runs of all methods were conducted first to identify the latest stagnation starting point of these methods. The limit of function calls is then set to 2~3 times the number of calls used at the latest stagnation starting point.



Problem Setting | Academic Runs | C

Objective ( ) Maximun (●) Minimum

Position 0 0 Randomly Reset

Search Increment	1	1
Mass	8	
Velocity	5	
Branch Momentum	20	
Maximum SubFlows	3	
Minimum Momentum	3.0	
Initial Flow Number	1	
Number of Bins	10	
Weight Capacity	160	

Precipitation

☑ Precipitation Operation

Interval Iterations	10
Precipitation Position Limit	25

Stop Criteria

Iterations	20000
Function Calls	100000

☐ Tradition Method ☑ Charting ☐ Repair

Execution: Reset | One Step | RunToEnd | 100%

☑ Color Each Block ☐ Show Legend ☐ Show Grids White

WFA Performance | Extra Information

Iterations	198
Number Of Flows	9
Average Flow Number	9.9596
Best Objective Value	2.190890
Found at Iteration	189
Known Optimum Value	0
Number of Precipitation Excutions	19
Number of Objective Function Calls	100,278

Repair

Best Solution So Far

```
Object 0 binID = 3 weight = 48
Object 1 binID = 4 weight = 49
Object 2 binID = 0 weight = 53
Object 3 binID = 7 weight = 18
Object 4 binID = 0 weight = 36
Object 5 binID = 6 weight = 10
Object 6 binID = 6 weight = 86
Object 7 binID = 9 weight = 77
Object 8 binID = 9 weight = 73
Object 9 binID = 0 weight = 20
Object 10 binID = 5 weight = 90
Object 11 binID = 5 weight = 40
Object 12 binID = 6 weight = 35
Object 13 binID = 3 weight = 50
Object 14 binID = 7 weight = 65
Object 15 binID = 4 weight = 99
Object 16 binID = 3 weight = 51
```

Save All Process | Save Best Solution

Figure 2: A snapshot of WFADOS

Numerical Test on a Designed BBP

This BPP has 30 objects with various weights to be packed into 10 bins. Therefore, $q = 30$ and $\xi = 10$ in WFA model. The weight capacity of each bin is 160 and the goal is to minimize the weight deviation between bins. Table 1 lists the weights of these objects in the global best packing result whose weight deviation is zero. Table 2 outlines the parameter settings of WFA for solving this problem. Several lengthy pilot runs were executed first for GA and WFA. Figures 3 and 4 are the objective function values obtained from WFA and GA, respectively. Notice that point A in Fig. 3 is regarded as the stagnation starting point which had made about 38,000 calls for the objective function. Stagnation starting point B in Fig. 4, in contrast, used about 36,000 calls. Therefore, the function call limit was set to 100,000 (about 2.5 times of 38,000) for both methods.

Table 1. A known optimal packing result whose weight deviation is zero

Bin ID	Objects in Weights				Bin ID	Objects in Weights				
1	48	49	53		6	35	50	65		
2	18	36	10	86	7	99	51			
3	77	73			8	20	21	22	13	74
4	20	90	40		9	71	79			
5	35	35	40	40	10	72	78			

Ten executions were conducted on the designed problem using GA and WFA individually. Obtained deviations were averaged and compared in Table 3. Note that the global optimal solution illustrated in Table 1 is comparatively unique, since the number of objects packed into a bin is small (around 3). No any execution in Table 3 hit the global optimum. However, the objectives (weight deviations) computed from WFA are in general smaller than those from GA. Specifically, the lowest of WFA (1.788) is smaller than that of GA (3.830). Table 4 illustrates the best packing result of WFA obtained in Run 5, whose weight deviation between bins is 1.788. In addition, the average number of bins that violate the capacity limit of WFA is lesser than GA as shown in Table 3.

Table 2. Parameter setting for WFA in solving the designed BPP

Initial Mass: M_0=8	Base Momentum: T =20
Initial Velocity: V_0 = 5	Number of Iterations for Precipitation: t =15
Subflow Number Limit: $\bar{n}$ =3	Maximal Number of Objective Function Calls =100000

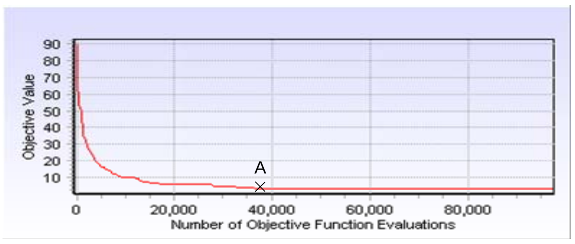


Figure 3. Objectives of a WFA pilot run versus the number of objective function calls

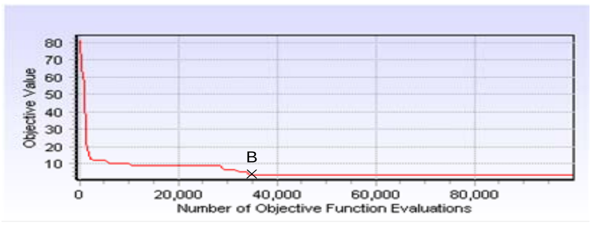


Figure 4. Objectives of a GA pilot run versus the number of objective function calls

The convergence trends of the objectives shown in Figs. 3 and 4 are typical responses from the executions of WFA and GA, respectively. At the beginning, the objective decreasing of GA is faster than that of WFA. However, the objective is leveled quickly at a poor level than the level of WFA. In contrast, the curve for WFA is slowly decreased but kept reducing steadily. As noted, since GA has more solution agents deployed than a single starting flow of WFA, the solution quantity improvement is effective in the beginning. In general, GA can perform extensive exploration on the solution space. However, without the aids of local searches, the solution quality of combinatorial optimization problems usually has room for improvements [7]. On the other hand, WFA steadily deploys solution agents to exploit the neighboring solutions and merges the duplicated ones. Whenever the flows are stuck in depressions, evaporation and precipitation operations are executed to relocate them to new positions. Consequently, the precipitation operation of WFA plays the role of a solution explorer.

Table 3. Comparison of the objectives obtained from WFA

Run	Weight Deviations		Number of Overloaded Bins	
	WFA	GA	WFA	GA
1	5.291	4.371	1	0
2	7.113	6.043	1	1
3	4.753	7.241	0	1
4	5.385	5.581	0	1
5	1.788*	9.201	0	2
6	7.912	3.830*	2	0
7	7.238	8.642	1	2
8	6.449	5.077	0	0
9	4.242	5.888	0	0
10	5.585	5.699	0	1
Average	**5.576**	**6.157**	**0.5**	**0.8**

Table 4. The best bin packing result of WFA, computed from Run 5

Bin ID	Objects in Weights	Total Weight	Bin ID	Objects in Weights	Total Weight
1	50, 99	149	6	71, 79	150
2	74, 78	152	7	77, 72	149
3	53, 20, 35, 40	148	8	49, 36, 35, 21, 13	154
4	86, 65	151	9	48, 18, 10, 73	149
5	20, 90, 40	150	10	40, 51, 22, 35	148

Solution exploitation is crucial in finding the global optima for discrete optimization problems. GA can only perform a limited exploitation on the solution space in contrast to its powerful exploration of crossover and mutation operations. Inferior solutions generated from crossover operations are quickly discarded and replaced with better ones, to promote and hasten the solution evolution. Fine grained neighboring searches are deficient in GA. In general, special designs of local searches are usually added in solving particular discrete optimization problems to enhance the solution quality. The presented flow splitting and moving operation of WFA focuses on solution exploitation and the precipitation focuses on exploration.

Effective solution convergence is a characteristic feature of GA. Although the mutation operation is designed to escape from converging to local optimum, mutated solutions with inferior qualities cannot easily survive from the selection operation of GA, which will maintain a size-fixed population of chromosomes. Therefore, the quality improvement using the mutation operation is not much prominent. On the other hand, when the precipitation operation of WFA generates multiple flows, no any selection mechanism is imposed to sacrifice the inferior solutions. Even though the newly pour-downed flow is inferior comparing to others, initial velocity assigned to the flow might drive it to the next iteration for improvements. Therefore, effective explorations can be conducted via the precipitation operation, the solution quality can be effectively improved iteration by iteration.

Numerical Tests on a Benchmark of OR-Lib

To further investigate the effectiveness of WFA, a tough BPP benchmark in OR-Lib was tested, whose file name is 38319.bpp. This benchmark has 120 objects with various weights to be packed into 48 bins. The weight total of these objects is 7078 and the weight capacity of bin is 150. Since the total weight capacity of the 48 bins is 7200 (=48 $\times$ 150), the weight allowance in object packing for each bin is only 2.54 (=(7200-7078)/48). Although there are 48^{120} solutions theoretically, few feasible solutions exist when the weight capacity constraint is imposed on bins.

Two optimization modes were tested: one was to minimize the weight deviation of the object-packed bins without considering the weight capacity constraint, and the other did impose this constraint. In this experiment, the first mode was tested using GA, EM, PSO, and WFA respectively to compare packing results between them. If the capacity constraint is imposed, the problem is highly constrained and repairing mechanisms are required for all methods. Since the repairing mechanisms cannot be fairly designed for all methods, only GA and WFA were tested in the second mode. Instead of implementing different repairing mechanisms, a heuristic adjustment operation is developed to modify infeasible solutions generated from both methods. Details will be explained later.

Similar to the previous experiment, pilot runs were executed first to identify a pertinent limit of objective function calls to serve as the stopping criterion for all of the methods under test. The limit was set to 500,000. The parameter settings of WFA were the same for both modes as shown in Table 5.

Table 5. Parameter settings for WFA in executing the benchmark.

Initial Mass: M_0 =8	Base Momentum: T =20
Initial Velocity: V_0 = 5	Number of Iterations for Precipitation: t =20
Subflow Number Limit: $\bar{n}$ =3	Maximal Number of Objective Function Calls =500,000

(1)*The Mode without Weight Capacity Constraint*

Methods of WFA, GA, PSO, and EM were tested in this mode. Since PSO and EM are mainly developed for solving real-numbered optimization problems (continuous optimization problems), a simple runoff operation was used to map real-numbered solutions to integer-numbered ones for BPPs. Five executions were conducted for each method and the weight deviations are listed and compared in Table 6.

Overall, the weight deviation obtained from WFA was the smallest. Notice that although implementation details are different among GA, PSO, and EM, they all perform lively exploration on the solution space. Agile solution explorations will easily discard inferior solutions and flock the size-fixed population of agents to the same solution. Therefore, the solution search is likely to converge to a local minimum and search stagnation happens. On the other hand, when a water flow appears in an inferior, comparing to others, yet promising location, the flow of WFA is not expelled out off the population. Moreover, when the flow flows to a promising area, the potential energy gain accelerates the flow to deploy more flow agents for solution exploitation. Conversely, when a flow enters an unfavorable location, the objective loss will deprive the kinetic energy of the flow and make the flow stagnated. No wasteful or redundant searches are executed. Conducting concurrent multiple flow searches without excluding others, WFA can efficiently and steadily explore and exploit the solution space. The solution search of WFA is not lively but steady and goal-driven.

Table 6. Comparison of the weight deviations obtained from four heuristic methods

Run	WFA	GA	PSO	EM
1	19.270	37.484	43.592	46.300
2	22.203	30.933	69.409	49.445
3	17.709	33.220	58.204	44.536
4	23.375	28.341	61.642	58.101
5	19.290	29.396	38.173	68.893
Average	**20.369**	**31.875**	**54.204**	**53.455**

(2) *The Mode with Weight Capacity Constraints*

This paper presents a solution adjustment operation to modify a bin packing result to be comply with the capacity constraint as much as possible. Whenever an infeasible packing result is generated, the proposed adjustment operation is executed to try to reduce the number of overloaded bins. The outcome of the adjustment operation is not guaranteed to be a feasible solution, but should be an improved one.

The adjustment operation is to recursively move the lightest object from the bin with the largest weight to the bin with the smallest weight, while the movement does not make the latter bin overloaded. Figure 5 illustrates an example of the adjustment operation, in which the weight capacity is 45. In the first iteration, the object of weight 23 is moved from bin 1 to bin 5, resulting in a weight of 37 for bin 5. Bin 1 is then not overloaded. In the second iteration, the object of weight 13 is moved from bin 7 to bin 4, resulting in a weight of 45 for Bin 4. In the third iteration, if the object of weight 25 is removed from bin 3 to bin 1, the cumulated weight of bin 1 will be 61 (=25+36), which violating the capacity constraint 45. Therefore, the adjustment operation is terminated and completed. Notice that even though the final solution is still infeasible (Bin 3 is still overloaded), the weight deviation is substantially reduced from the original 12.64 to 4.61.

Iteration 1		**Iteration 2**		**Iteration 3**	
Bin	Weight	Bin	Weight	Bin	Weight
1 {36, **23**}	**59** max	1 {36}	*36*	1 {36}	**36** min
2 {38, 7}	45	2 {38, 7}	45	2 {38, 7}	45
3 {25, 25}	50	3 {25, 25}	50	3 {25, **25**}	**50** max
4 {18, 14}	32	4 {18, 14}	**32** min	4 {18, 14, *13*}	*45*
5 {12, 2}	**14** min	5 {12, 2, *23*}	*37*	5 {12, 2, 23}	37
6 {45}	45	6 {45}	45	6 {45}	45
7 {41, 13}	54	7 {41, **13**}	**54** max	7 {41}	*41*
8 {21, 22}	43	8 {21, 22}	43	8 {21, 22}	43
9 {31, 12}	43	9 {31, 12}	43	9 {31, 12}	43
10 {36}	36	10 {36}	36	10 {36}	36
STDEV: 12.64		STDEV: 6.87		STDEV: 4.61	

Figure 5. An example of the adjustment operation applied to an infeasible solution

Since the performances of PSO and EM are comparatively inferior to GA and WFA, only GA was selected to compare with WFA in solving this benchmark, where the adjustment operation was used for modifying the overloaded results. In this experiment, 10 runs were executed for each method. The weight deviations and the numbers of bins violating the weight constraint are listed in Table 7. Table 8 list the best packing result from Run 8 of WFA.

Currently the optimum solution to this benchmark still remains undiscovered. In addition, whether there is a feasible solution is still unknown. Nevertheless, the results from WFA are better than those from GA. The possible reasons have been discussed in the previous example. Comparing the results shown in

Tables 6 and 7, the average weight deviation is reduced from 20.369 to 14.449 for FWA with the help of the adjustment operation. GA has a larger reduction from 31.875 to 19.968, since its solutions are in general inferior to FWA.

5. CONCLUSIONS AND SUGGESTIONS

The WFA presented in this paper is a novice heuristic approach to solving discrete optimization problems. The method has adopted several characteristic behaviors of water flows. The objective surface defined in the solution space is mapped as a landform and the solution agents are modeled as water flows. Flow splitting, moving, and merging operations are designed to fork, move, and join solution agents. The changing number of agents is a primary difference from other population-based methods. The solution exploration conducted by the precipitation operation and the solution exploitation conducted by moving the flow to neighboring solutions make the solution search of WFA steady and persistent.

Table 7. Computation results of the benchmark from GA and WFA with adjustment operation applied

Run	Weight Deviations		Number of Overloaded Bins	
	WFA	GA	WFA	GA
1	13.489	15.233	18	24
2	14.241	20.055	19	24
3	15.036	22.259	18	26
4	15.837	17.353	21	25
5	15.787	24.950	18	25
6	15.444	23.652	20	25
7	13.414	14.988*	19	31
8	11.311*	24.712	17	24
9	16.349	19.925	17	21
10	13.579	16.552	20	25
Average	**14.449**	**19.968**	**18.7**	**25.714**

The numerical tests have shown that the WFA performed well in solving BPPs. Future efforts can be worked on sequencing problems, including the TSP and JSP. On the other hand, a general WFA for continuous optimization problems is worth further development, where a different flow moving operation is to be expected. Instead of moving a flow or subflows to the neighboring solutions, the velocity information should be used to determine the new locations. Although WFAs for solving other optimization problems are still under developing, the heuristic approach deserves more attention on the concept of using a size-not-fixed population of agents and the techniques for balancing solution exploration and exploitation.

Table 8. The best solution obtained from Run 8 of WFA

Bin ID	Objects in Weights	Total Weigh t	Bin ID	Objects in Weights	Total Weight
1	84,80	164	25	49, 38, 35	122
2	49, 44, 78	171	26	92, 57	149
3	70, 79	149	27	74, 30, 45	149
4	85, 60	145	28	27, 36, 37, 43	143
5	28, 81, 39	148	29	98, 43	141
6	57, 46, 33	136	30	26, 69, 43	138
7	25, 20, 98	143	31	69, 41, 58	168
8	90, 32, 24	146	32	50, 78	128
9	55, 84	139	33	45, 84	129
10	62, 94	156	34	69, 30, 55	154
11	85, 62	147	35	73, 74	147
12	87, 83	170	36	57, 98	155
13	42, 58, 41	141	37	83, 70	153
14	27, 93, 23	143	38	33, 84, 29	146
15	67, 42, 25	134	39	59, 96	155
16	80, 60	140	40	43, 73, 32	148
17	38, 64, 42	144	41	86, 23, 42	151
18	44, 36, 66	146	42	41, 91	132
19	82, 76	158	43	55, 91	146
20	93, 71	164	44	78, 84	162
21	38, 47, 58	143	45	36, 46, 30, 49	161
22	79, 39	118	46	57, 96	153
23	73, 80	153	47	82, 73	155
24	33, 42, 70	145	48	72, 78	150

REFERENCES

1. Birbil, S. I. and S. C. Fang, "An electromagnetism-like mechanism for global optimization," *Journal of Global Optimization*, **25**, 263-282 (2003).
2. Dougherty, D. E. and R. A. Marryott, "Optimal Groundwater Management," *Water Resources Research*, **27**, 2493-2508 (1991).
3. Falkenauer, E., "A hybrid grouping genetic algorithm for bin packing," *Journal of Heuristics*, **2**, 5-30 (1996).
4. Gen, M. and R. Cheng, *Genetic Algorithms and Engineering Design*, Wiley, New York (1997).
5. Glover, F. and M. Laguna, *Tabu Search*, Springer, (1998).
6. Glover, F., E. Taillard and E. Taillard, "A user's guide to tabu search," *Annals of Operations Research*, **41**, 11-28 (1993).
7. Hansen, P., N. Mladenovic, V. Stefan and M. Silvano, "An Introduction to Variable Neighborhood Search," in S. Voss, S. Martello, I. H. Osman and C. Roucairol (eds), *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer Acadenics, Boston, 433-458 (1999).
8. Holland, J. H., *Adaptation in Natural and Artificial System*, Ph.D. Thesis, University of Michigan (1975).
9. Kennedy, J., "Stereotyping: improving particle swarm performance with cluster analysis," *Proceedings of IEEE International Conference on Evolutionary*

Computing, Jul 16-19, California, USA, 1507-1512 (2000).

10. Kennedy, J. and R. Eberhart, "Particle swarm optimization," *Proceedings of IEEE International Conference on Neural Networks*, Nov. 27 - Dec. 1, Perth, Aust, 1942-1948 (1995).
11. Moore, J. and R. Chapman, "Application of particle swarm to multiobjective optimization," *Report of Department of Computer Science and Software Engineering*, Auburn University (1999).
12. OR-Lib, Benchmark Library, Available online at: http://mscmga.ms.ic.au.uk/info.html (accessed April 1, 2006).
13. Yang, F. C., C. Y. Kang and Y. H. Hung, "Electromagnetic attraction and repulsion simulated techniques-based optimization framework and software Systems," *Proceedings of the 11th Annual International Conference on Industrial Engineering*, Oct. 24-27, Nagoya, Japan, 1338-1343 (2006).
14. Yang, F. C. and C. L. Wei, "Electromagnetic-like mechanism based discrete optimization algorithm for object sequencing problems and object grouping problems," *Proceedings of the 7th Asia Pacific Industrial Engineering and Management Systems Conference*, Nov 17-20, Bangkok, Thailand, 136-144 (2006).
15. Shi, Y. and R. C. Eberhart, "Fuzzy adaptive particle swarm optimization," *Proceedings of the 2001 Congress on Evolutionary Computation*, May 27-30, Seoul, Korea, 101-106 (2001).
16. Zheng, C. and P. P. Wang, "Parameter structure identification using tabu search and simulated annealing," *Advances in Water Resources*, **19**, 215-224 (1996).

ABOUT THE AUTHORS

Feng-Cheng Yang is currently an associate professor in the Graduate Institute of Industrial Engineering at National Taiwan University. He received the Bachelor's degree in Mechanical Engineering from National Taiwan University, in 1980, and the Ph.D. degree in Mechanical Engineering from the University of Iowa, USA, in 1991. His present research interests include various heuristics techniques for solving industrial optimization problems. Currently, his focus is the newly developed WFAs for various optimization problems. He has developed several software systems covering various heuristics: ant colony optimization algorithms, electromagnetism-like mechanism, genetic algorithms, wrapped-round self-organizing maps, etc. These systems are developed to facilitate academic research and teaching. Various benchmark problems frequently used for new algorithm verification are included in these systems. Readers are welcome to download these systems for their interests. The URL is http://www.ie.ntu.edu.tw/.

Yuan-Peng Wang is currently an employee of Acer Inc., Taiwan. He received the Master degree from the Graduate Institute of Industrial Engineering, National Taiwan University, in 2006.

(Received December 2006; revised March 2007; Accepted May 2007)

物件分群之仿水流優化演算法

楊烽正*、王元鵬
國立台灣大學工業工程學研究所
106 台北市大安區羅斯福路四段一號

摘要

本文提出一個新的啓發式演算法名爲「仿水流優化演算法」(Water Flow-Like Algorithm，WFA)以求解物件分群問題。WFA的求解機制是模仿水流在地理空間流動的物理特性。地理空間中的水流停駐的位置被應對爲優化問題求解空間的一個解。一股水流即代表一個解，水流即是解的代理人。水流會受地心引力影響不斷地向低處流動。流動的過程中隨著地理空間的變化會有分流、匯流等現象。部分水流會蒸發至大氣中形成水氣，降水後繼續流動。自然界的水流泰半會流經地理空間中的最低點。WFA在解搜尋過程中透過分流、匯流、蒸發、降水四個演化作業調整水流代理人的數量，兼顧節空間的探索(Exploration)和探究(Exploitation)，避免搜尋過度或搜尋不足。本文展示的WFA係以離散優化問題爲求解對象設計演化作業細節。此外也將針對物件分群優化問題中的裝箱問題(Bin Packing Problem, BPP)進行求解驗證。求解結果與遺傳演算法、仿電磁吸斥優化演算法、及粒子群演算法比較以驗證求解效能。範例測試則針對複雜度高的自訂範例和OR-Library內的標竿問題求解，並與其他方法比較。結果顯示，WFA在求解BPP問題上有優於其他啓發式演算法的趨勢。

關鍵詞：啓發式演算法，仿水流優化演算法，匯流，分流，降水
(*聯絡人: iefcyang@ntu.edu.tw)