



# Cooperative group optimization with ants (CGO-AS): Leverage optimization with mixed individual and social learning



Xiao-Feng Xie<sup>a,\*</sup>, Zun-Jing Wang<sup>a,b</sup>

<sup>a</sup> WIOMAX LLC, Pittsburgh, PA 15213, United States

<sup>b</sup> Department of Physics, Carnegie Mellon University, Pittsburgh, PA 15213, United States

## ARTICLE INFO

### Article history:

Received 12 February 2016  
Received in revised form 5 September 2016  
Accepted 8 November 2016  
Available online 18 November 2016

### Keywords:

Ant systems  
Cooperative group optimization  
Traveling salesman problem  
Global optimization  
Group intelligence  
Population-based methods  
Socially biased individual learning

## ABSTRACT

We present CGO-AS, a generalized ant system (AS) implemented in the framework of cooperative group optimization (CGO), to show the leveraged optimization with a mixed individual and social learning. Ant colony is a simple yet efficient natural system for understanding the effects of primary intelligence on optimization. However, existing AS algorithms are mostly focusing on their capability of using social heuristic cues while ignoring their individual learning. CGO can integrate the advantages of a cooperative group and a low-level algorithm portfolio design, and the agents of CGO can explore both individual and social search. In CGO-AS, each ant (agent) is added with an individual memory, and is implemented with a novel search strategy to use individual and social cues in a controlled proportion. The presented CGO-AS is therefore especially useful in exposing the power of the mixed individual and social learning for improving optimization. The optimization performance is tested with instances of the traveling salesman problem (TSP). The results prove that a cooperative ant group using both individual and social learning obtains a better performance than the systems solely using either individual or social learning. The best performance is achieved under the condition when agents use individual memory as their primary information source, and simultaneously use social memory as their searching guidance. In comparison with existing AS systems, CGO-AS retains a faster learning speed toward those higher-quality solutions, especially in the later learning cycles. The leverage in optimization by CGO-AS is highly possible due to its inherent feature of adaptively maintaining the population diversity in the individual memory of agents, and of accelerating the learning process with accumulated knowledge in the social memory.

© 2016 Elsevier B.V. All rights reserved.

## 1. Introduction

Ants are extremely successful in evolution of intelligence. As shown by socio-biologists [31], although each ant only has a minuscule brain, nontrivial primary components of intelligence in the collective context have been encoded in their navigational guidance systems using multiple simple information sources. For example, ants can communicate with others through indirect means of the pheromone trails that they deposited in their environment [17]. The collective foraging behavior and strong exploitation capability in ant colonies has inspired the invention of various ant system (AS) algorithms [5,19,20]. Typical AS includes Ant Colony Optimization (ACO) [19], AS with ranking (AS<sub>rank</sub>) [7], and MAX-MIN ant system (MMAS) [48]. In addition, there are diverse hybrid forms of AS with other optimization algorithms, such as PSO-ACO-3Opt [38],

ACO-ABC [28], and FOGS-ACO [46]. Among the existing algorithms of AS, the usage of pheromone trails in natural ants has attracted a broad research interest [4,8–10,50]. Pheromone trails of ants have now been adopted as a paradigm by computational research communities to illustrate the emergence in self-organization [17]. Though ignoring individual memory of ants, the algorithms of AS—mainly by the construction and usage of pheromone trails in natural ants—has displayed a remarkable optimizing capability, and has been applied with a great success to a large number of computationally hard problems, such as the traveling salesman problem (TSP) [41,45,48], vehicle routing problems [7], and mixed-variable problems [35].

Natural ants however build their intelligence with both social and individual learning. Socio-biologists have found that individual route memory of ants [23,50] plays a significant role in guiding the foraging of many natural ant species [4,11,29,37,47]. A fairly great amount of natural ant species [14,27] uses both collective pheromone trails and individual route memory in their navigational guidance systems, though it remains unclear how ants leverage their search in complex environments with such

\* Corresponding author.

E-mail addresses: [xie@wiomax.com](mailto:xie@wiomax.com) (X.-F. Xie), [wang@wiomax.com](mailto:wang@wiomax.com) (Z.-J. Wang).

an integrated usage mixing the two memories which correspond respectively to their social and individual learning.

In the present work, we aim to study the benefit using a mixed individual and social learning in AS systems. Since the existing AS algorithms have demonstrated the optimization power of social learning, what leverage can be gained in optimization by merely adding individual learning? For a better performance, how to distribute the social and individual learning if one mixes and uses them together as an integrated form of intelligence? In this sense, our computational experiments do not aspire to providing complete comparison with state of the art algorithms using various instances (e.g. across various algorithms and optimization problems). Rather, we attempt to understand the leveraging aspects in optimization from simply adding and mixing individual learning into its original solely-social-learning version of AS systems. This is because AS is a succinct but intrinsic model for understanding the role of learning. Given how technically involved an upgraded-learning-induced optimization improvement of an algorithm is, a question of considerable practical relevance is: How to effectively integrate individual learning into the system? The question is certainly not trivial. Realization of a mixed learning in an integrated form requires an algorithm being a more complex system which encompasses interactions among multiple memories (individual and social memories) and behaviors. A fundamental and effective support must be provided for maintaining the fast self-organized processes in such a mixed learning.

In this study, we will approach the question with a specific framework, the cooperative group optimization (CGO) framework [54]. CGO was presented based on the nature-inspired paradigm of group problem solving [18,22,25,34,42,44,49], to explore high-quality solutions in the search landscape [33] of the problem to be solved. The agents of CGO not only exploit in a parallel way using diverse novel patterns [42] preserved in their individual memory [21,24], but also cooperate with their peers through the group memory [16,18]. This means that each agent of CGO possesses a search capability through a mix of both individual and social learning [6,22,49]. Therefore, we use CGO to implement a generalized AS, called CGO-AS. The presented CGO-AS is especially suitable to realize the cooperative search using both individual route memory and pheromone trails, and to reproduce a navigational guidance system of natural ants. Moreover, CGO-AS provides an algorithmic implementation of AS systems in a generalized form of group problem solving, which is commonly used in advanced social groups including human groups. This is important, as the generalization of AS not only enables us to find the advanced strategies used by ants to strengthen their optimization, but also allows us to observe the potential nontrivial factors which might contribute to the primary form of group intelligence from the low-level cognitive colonies. We use the TSP [45], a well-known computationally hard problem, as the testing benchmark of performance for the comparison between CGO-AS and other existing AS systems as well as some recent published algorithms [38,43,46,48,55].

The remainder of the paper is organized as follows. In Section 2, we describe the studies of the navigational guidance components and systems used by natural ants in their foraging to digest the fundamental features of their learning behaviors which motivated AS systems and CGO-AS of this work. In Section 3, we briefly introduce AS systems and one representative example, the MAX-MIN version. In Section 4, we outline CGO framework. In Section 5, we present our CGO-AS imitating the natural ants with both social and individual learning, and describe how to use it for solving the TSP. In Section 6, we present our experimental results showing the performance of the proposed CGO-AS approach, and discuss its features. Finally, we summarize our work.

## 2. Real-world ant navigation

Natural ants are important models for understanding the role of learning in evolution of intelligence and in the improvement of optimization technologies [20,48,50]. Ant workers have miniature brains but often striking navigational performance and behavioral sophistication as individuals of socially complex cognitive colonies [50]. Understanding the robust behavior of ants which solve complex tasks unveils parsimonious mechanism and architecture for general intelligence and optimization. Here we will first briefly review the usage of pheromone trails by ants in their foraging, which provided the foundation of existing AS systems [20,48]. Next, we will then describe the usage of individual memory by ants and their more advanced navigational guidance systems, which inspires the realization of CGO-AS in this work.

### 2.1. Pheromone trails

Many ant species can form and maintain pheromone trails [15,40], even the volatile ones. The study of the fire ants has showed that pheromone trails provide feedback to ants for organizing the massive foraging at a colony level [31]. Successful foragers deposit pheromone on their return trails to the nest, resulting in the effective trails strengthened since more workers add pheromone to it. On the contrary, the trail decays if its food runs out, because foragers refrain from reinforcing the trail and the existing pheromone of the trail evaporates. Pheromone trails provide ants a long-term memory of previously used trails, as well as a short-term attraction to recent rewarding trails [32].

Concerning pheromone trails of ants, a global adaptive process arises from the activities of many *agents* responding to local information in shared environments [17]. The performance is achieved through the social learning of ants, with which ants have mutual interactions via their pheromone trails. Computational models of ant systems [20,48] have showed how ant workers could cooperate together through social learning via the pheromone trails, which exhibits an impressive optimization capability on some complex problems, such as the finding of short paths in the TSP.

### 2.2. Route memory

Ants also navigate using vectors and landmark-based routes [23,50], as shown in many species, for example, the wood ant (*Formica rufa*) [29], the tropical ant (*Gigantiops destructor*) [37], the Australian desert ant (*Melophorus bagoti*) [47], and the North African desert ant (*Cataglyphis fortis*) [4,11]. In foraging, individual ants can obtain their routes by initial navigational strategies [11,50], can put their innate responses to landmarks [50], and can also memorize early routes with their increasing experience [13,52]. These behaviors attribute to the individual learning ability of ants, which is fundamental for evolving the advanced forms of general intelligence [24].

A great deal of flexibility has been observed in the individual learning of ants on their route navigation [31]. Ants can steer by visual *landmarks* in their route navigation [12]. They instruct others when they recall particular steering cues [51]. Ants can also learn path segments in terms of the associated *local vectors* that connect between landmarks [11]. In addition, ants can memorize multiple routes [47], and can even steer the journeys that consist of separate path segments. Information combined from all experienced path segments may be used by ants as a memory network [53] to determine their familiar headings on given landmarks. Route memory often plays a significant role in guiding ants during their foraging activities.

### 2.3. Navigational guidance system

Ants integrate information from multiple sources in their navigational guidance systems [4,8–10,50] in order to efficiently search the paths between goals. For example, some ants [14,27] use both pheromone trails and route memory in foraging. Notice that pheromone trails and route memory are respectively corresponding to the social and private individual information of ants that support their social and individual learning. Pheromone trails may cover more foraging paths by encoding the collective experiences of

where  $\mathcal{N}_i$  is the candidate set of cities which the ant has not visited yet, and  $\alpha$  and  $\beta$  are two setting parameters which control the relative importance of the pheromone trail and heuristic information. By default,  $\alpha = 1$  and  $\beta = 2$ . For the TSP,  $\eta_{ij}$  is a function of the edge length, i.e.,  $\eta_{ij} = 1/d_{ij}$ . Normally, the selection in Line 3 (see Algorithm 1) is augmented with the candidate set of length 20 which contains the nearest neighbors [48] to reduce the computational cost.

**Algorithm 1.** Construct a tour by an ant using the pheromone matrix  $\Psi$

```

Require: The pheromone matrix  $\Psi$ 
1:  $\mathcal{N} = \{1, \dots, N\}$ ;  $i = \text{RND}(\mathcal{N})$ ;  $\mathcal{N}_i = \mathcal{N} \setminus \{i\}$ ;  $\pi_{[1]} = i$  // The cost matrix  $D$  is a default input
2: for  $n = 2$  to  $N$  do // From a randomly selected city
3:   Select the next city  $j \in \mathcal{N}_i$  with the probability  $p_{ij}^{(t)}$ , using Eq. 1 with  $\Psi$  and  $D$ 
4:    $\pi_{[n]} = j$ ;  $\mathcal{N}_j = \mathcal{N}_i \setminus \{j\}$ ;  $i = j$  // Move to the next city  $j$ 
5: end for
6: return  $\pi = (\pi_{[1]}, \dots, \pi_{[N]})$  // Return the new tour
    
```

ants, but route memory is often more accurate in information than pheromone trails, even limited by the minuscule brains of ants.

Natural ant colonies often use an integrated mixed learning in their foraging system, where route memory and pheromone trails combine together to a synergistic information cascades cooperatively providing an effective and efficient guidance over various foraging conditions. Route memories maintain a diversity of the high-quality information learned from individual experience of each ant, while pheromone trails provide a stability of the high-quality routes learned from all ants and over time [12]. The understanding on the real-world navigational guidance system of ants motivated us to present, implement and test CGO-AS system in this work.

### 3. Ant systems for the TSP

AS [20] is a class of optimization algorithms inspired by the emergent search behavior using *pheromone trails* [15] in natural ants [26]. Though different optimization problems [7,35,45,48] have been solved with AS variants, the TSP is normally considered as a testing benchmark of ant navigation.

The TSP [45] can be described as a complete graph with  $N$  nodes (or cities) and a cost matrix  $D = (d_{ij})$ , in which  $d_{ij}$  is the length of an edge  $(i, j)$  that connects between cities  $i$  and  $j$ , where  $i, j \in [1, N]$ . The study here only concerns the symmetric TSP, which has  $d_{ij} = d_{ji}$  for the edges. Each potential solution is a Hamiltonian tour  $\pi = (\pi_{[1]}, \dots, \pi_{[N]})$ , which passes through each node once and only once, and its evaluation value  $f(\pi)$  is the total length of all edges in the tour. The optimization objective is to find a tour with the minimal evaluation value.

In AS, there are a colony of  $K$  artificial ants, where all ants search using a *pheromone matrix*  $\Psi = (\tau_{ij})$ , in which  $\tau_{ij}$  describes the pheromone trail from city  $i$  to city  $j$ . The system runs in total  $T$  iterations. At each iteration  $t$ , each ant builds its tour  $\pi^{(t)}$  in an iterative way. As shown in Algorithm 1, starting from a randomly selected city as the current city  $i$ , each ant chooses the next city  $j$  to go with a probability biased by the pheromone trail  $\tau_{ij}^t$ , *heamount*( $t$ ) and by a locally available heuristic information  $\eta_{ij}$  present on the connecting edge  $(i, j)$ , and continues this process till a tour is built. When an ant is at city  $i$ , the selection probability of the ant to city  $j \in \mathcal{N}_i$  is described as [20]:

$$p_{ij}^{(t)} = \frac{[\tau_{ij}^{(t)}]^\alpha [\eta_{ij}]^\beta}{\sum_{l \in \mathcal{N}_i} [\tau_{il}^{(t)}]^\alpha [\eta_{il}]^\beta}, \quad (1)$$

After all ants have constructed their tours, pheromone is updated on all edges as follow:

$$\tau_{ij}^{(t+1)} = \rho \cdot \tau_{ij}^{(t)} + \sum_{k=1}^K \Delta \tau_{ij}^{(k,t)}, \quad (2)$$

where the parameter  $\rho \in [0, 1)$  is the trail persistence from evaporation,  $\Delta \tau_{ij}^{(k,t)} = 1/f(\pi^{(k,t)})$  is the amount of pheromone which the ant  $k$  puts on the edge  $(i, j)$  under the condition that the edge belongs to the tour  $\pi^{(k,t)}$  done by the ant  $k$  in the iteration  $t$ . By default,  $\rho = 0.5$ .

In Eq. (2), evaporation mechanism enables the system to “forget” unuseful edges over time, and a greater amount of pheromone is allocated to the shorter tours. In Eq. (1), the selection probability is achieved from a combination of the global heuristic cue (of tour length) from the pheromone trail  $\tau_{ij}(t)$  and the local heuristic cue (of edge length) from the heuristic information  $\eta_{ij}$ . Edges which are contained in the shorter tours will receive more pheromone and thus will be chosen by ants with higher probabilities in future iterations.

The ants in AS do not possess long-term individual memory. Rather, pheromone matrix  $\Psi$  plays the role of a long-term social memory distributed on the edges of the graph, which is iteratively modified by ants to reflect their experience accumulated in solving the problem. This allows an indirect form of learning called *stigmergy* [19].

#### 3.1. MAX-MIN ant system (MMAS)

MAX-MIN ant system (MMAS) [48] is one of the best performing ant systems. It has been specifically developed for achieving a better performance by the combination between an improved exploitation of the best solutions found in search and an effective mechanism which leads to the choice probabilities avoiding early search stagnation.

MMAS differs in two key aspects from the original AS. First, in order to impose strong exploitation on the best solutions found in search, after each iteration, only one ant deposits pheromone on the best solution, either in the current iteration (*iteration-best* solution  $\pi_{ib}^{(t)}$ ) or from the beginning (*best-so-far* solution  $\pi_{gb}^{(t)}$ ). Second, in order to prevent a search from stagnation, the range of possible pheromone trails is limited within an interval  $[\tau_{\min}, \tau_{\max}]$ . The pheromone trails are initialized to be  $\tau_{\max}$  to achieve a higher exploration of solutions at the beginning of the search.

The values of  $\tau_{\max}$  and  $\tau_{\min}$  are respectively defined as [48]:

$$\tau_{\max}^{(t)} = \frac{1}{(1 - \rho)f(\pi_{\text{gb}}^{(t)})}, \quad (3)$$

$$\tau_{\min}^{(t)} = \frac{\tau_{\max}(p_{\text{best}}^{-1/N} - 1)}{n/2 - 1}, \quad (4)$$

where  $f(\pi_{\text{gb}}^{(t)})$  is the evaluation value of the best-so-far solution  $\pi_{\text{gb}}^{(t)}$  at the iteration  $t$ ,  $\rho$  is the trail persistence,  $p_{\text{best}} \in [(N/2)^{-N}, 1]$  is a setting parameter. If  $p_{\text{best}} = 1$ , then  $\tau_{\min} = 0$ . The value of  $\tau_{\min}$  increases as  $p_{\text{best}}$  decreases. By default,  $p_{\text{best}} = 0.05$ .

#### 4. Cooperative group optimization (CGO)

CGO is an optimization framework based on the nature-inspired paradigm of group problem solving [54]. With CGO, optimization algorithms can be represented in a script form using embedded search heuristics (ESHs) with the support of memory protocol specification (MP-SPEC) for the group of agents, using a toolbox of knowledge components. CGO has been used to describe some existing algorithms and realize their hybrids on solving numerical optimization problems.

Fig. 1 gives a simplified version of CGO used in this paper for CGO-AS. The framework consists of a group of  $N$  agents, an *interactive center* (IC), and a *facilitator*. It runs iteratively as a Markov chain in total  $T$  learning cycles.

The components in Fig. 1 are defined with a list of acronyms as:

$F_P$ : The optimization problem to be solved

$F_R$ : The internal representation of  $F_P$  in the form of a *search landscape*

$\pi_{\text{gb}}$ : the *best-so-far* solution found for  $F_P$

$M_A$ : The *private memory* of each agent, which can be accessed and is updated by the agent

$M_{BA}$ : A buffer for storing the chunks generated by the agent for updating  $M_A$  in each cycle

$B_{UA}$ : The  $M_A$ -*updating* behavior to update  $M_A$  using the information (chunks) in  $M_{BA}$

$M_G$ : The *generative buffer* of each agent, which store newly generated information in each cycle

$M_S$ : The *social memory* in IC, which can be accessed by the agents and is maintained by IC

$M_{BS}$ : A buffer for collecting information (chunks) generated by the agents for updating  $M_S$

$B_{US}$ : The  $M_S$ -*updating* behavior to update  $M_S$  using the chunks in  $M_{BS}$

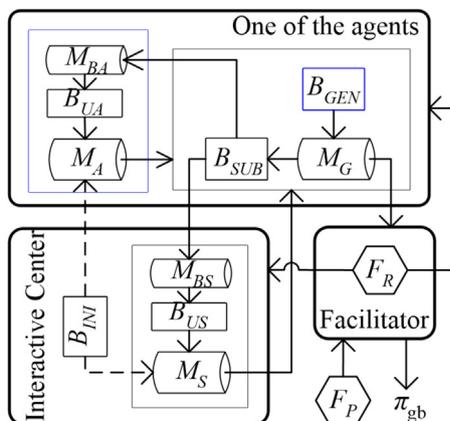


Fig. 1. Simplified CGO framework for CGO-AS.

$B_{INI}$ : The *initializing* behavior to initialize information in  $M_A$  of the agents and  $M_S$  of the IC

$B_{GEN}$ : The *generating* behavior of each agent, which generates new chunks into  $M_G$  of the agent, using the mixed information (chunks) from  $M_A$  of the agent and  $M_S$  in IC, in each cycle

$B_{SUB}$ : The *submitting* behavior of each agent, which submits the chunks in  $M_G$  to  $M_{BA}$  of the agent and to  $M_{BS}$  in IC, in each learning cycle

##### 4.1. Facilitator

The facilitator manages basic interfaces for the optimization problem  $F_P$  to be solved. An essential landscape for the problem can be represented as a tuple  $\langle S, R_M, AUX \rangle$ .  $S$  is the *problem space*, in which each *state*  $\pi \in S$  is a potential solution.  $\forall \pi_{(a)}, \pi_{(b)} \in S$ , the *quality-measuring* rule ( $R_M$ ) measures the *quality* difference between them. If the quality of  $\pi_{(a)}$  is better than that of  $\pi_{(b)}$ , then  $R_M(\pi_{(a)}, \pi_{(b)})$  returns TRUE, otherwise it returns FALSE.  $AUX$  contains all auxiliary components associated with the structural information of the problem.

The facilitator has two basic roles. The first role is to update the *best-so-far* solution  $\pi_{\text{gb}}^{(t)}$ , by storing the better-quality state between  $\pi_{\text{gb}}$  and each newly generated state using the  $R_M$  rule. The second role is to provide a *search landscape*, i.e.,  $F_R = \langle S, R_M, AUX \rangle$ , which includes all heuristic cues of the problem that is useful for reaching the high-quality states.

##### 4.2. Agents and IC

Searching on the search landscape is performed by agents with the support from IC. The general solving capability arises from the interplay between *memory* ( $M$ ) and *behavior* ( $B$ ) [1] owned by these entities. The actual implementation is flexibly defined by a symbolic *script* over a *toolbox* of knowledge element instances.

###### 4.2.1. Memory and behavior

Each memory [24] contains a list of cells storing basic declarative knowledge elements, called *chunks* ( $CH$ ) [1], which are associated with the information in the search landscape of  $F_R$ . During a runtime, each memory can be only updated by its owner. Each behavior, performed by its owner, applies *rule(s)*( $R$ ) to interact with some chunks in memory during a learning process.

There are three essential memories in CGO framework. IC maintains a social memory ( $M_S$ ). Each agent possesses a private memory ( $M_A$ ) and a generative buffer ( $M_G$ ).  $M_A$  and  $M_G$  can be accessed only by its owner, while  $M_S$  can be accessed by all agents. Both  $M_S$  and  $M_A$  are long-term memory (LTM) to hold chunks over learning cycles, while  $M_G$  is a buffer for new chunks and will be cleared at the end of each learning cycle.

IC holds two basic behaviors. The *initializing* behavior ( $B_{INI}$ ) is used to initialize chunks in  $M_A$  of the agents and  $M_S$  of the IC. IC also holds a buffer  $M_{BS}$  for collecting chunks from the agents. The  $M_S$ -*updating* behavior ( $B_{US}$ ) updates  $M_S$  by using the chunks in the buffer  $M_{BS}$ .

The search process for solving  $F_R$  is performed by agents. Each agent has the following basic behaviors: (a) The *generating* behavior ( $B_{GEN}$ ) can generate new chunks into a generative buffer ( $M_G$ ), using the chunks in both its  $M_A$  and  $M_S$  in IC; (b) the *submitting* behavior ( $B_{SUB}$ ) is used to submit chunks in  $M_G$  to  $M_{BA}$  of the agent and to  $M_{BS}$  in IC; (c) The  $M_A$ -*updating* behavior ( $B_{UA}$ ) is applied to update  $M_A$  using elements obtained in  $M_{BA}$ ; and (d) The state(s) in  $M_G$  are extracted and exported to facilitator as the candidates for potential solution(s). In each learning cycle,  $B_{GEN}$  is performed at first, and the newly generated elements

in  $M_G$  are processed afterwards by agents with the other three behaviors.

#### 4.2.2. Script representation

For agents and IC, the essential search loop is driven by *embedded search heuristic* (ESH) with the support of *memory protocol specification* (SPEC-MP). SPEC-MP is used to define how chunks will be initialized and updated in  $M_A$  of each agent and  $M_S$  of IC, given that chunks are newly generated in  $M_G$  of the agents, and we need to maintain the consistency of the interactions between memory and behavior. By defining the part to generate new chunks in  $M_G$  using the chunks in  $M_A$  of each agent and  $M_S$  of IC, each ESH is able to close the search loop. Therefore, each ESH can be seen as a stand-alone algorithm instance with a solid low-level support of SPEC-MP in the framework of CGO.

SPEC-MP contains a table of *memory protocol rows*, where each row contains five elements, i.e.,  $\langle ID_M, CH_M, R_{IE}, R_{UE}, CH_U \rangle$ .  $ID_M \in \{M_A, M_S\}$  refers to a long-term memory.  $CH_M$  is a unique chunk in the memory  $ID_M$ . Each row thus refers to the chunk  $CH_M$  in the memory  $ID_M$ .  $CH_U$  is a chunk in  $M_A$  and  $M_G$ . Each *elemental initializing* ( $R_{IE}$ ) rule is used to output one chunk  $CH_{(I)}$  for initializing  $CH_M$ . Each *elemental updating* ( $R_{UE}$ ) rule updates the chunk  $CH_{(M)}$  by taking two inputs ( $CH_{(M)}$ ,  $CH_{(U)}$ ). SPEC-MP can be split into two subtables, SPEC-MP<sub>A</sub> and SPEC-MP<sub>S</sub>, where their  $ID_M$  are respectively  $M_A$  and  $M_S$ . Notice the fact that there are multiple agents but only one IC. Corresponding to  $ID_M$  in  $M_A$  and  $M_S$ , the types for  $CH_{(I)}$  of  $R_{IE}$  are respectively  $\$CH_M$  and  $CH_M$ , the types for  $CH_{(U)}$  of  $R_{UE}$  are respectively  $CH_M$  and  $\$CH_M$ , and the types for  $CH_{(M)}$  of  $R_{UE}$  are both  $CH_M$ . Here  $\$CH$  means a set of chunks of the type  $CH$ .

layer using SPEC-MP and ESHs to call the instances in CGO toolbox.

#### 4.3. Execution process

Algorithm 2 gives the essential process for executing a single ESH with the support of SPEC-MP in the framework of CGO, where the working module (entity), the required inputs, and the outputs or updated modules are provided. In Line 1,  $F_P$  is formulated into the form of a static search landscape  $F_R = \langle S, R_M, AUX \rangle$ . In Lines 2 and 3, all long-term memories used by the agents and IC are initialized by using  $B_{INI}$ , based on SPEC-MP. After the initialization, the framework of CGO runs in the form of iterative learning cycles, where each learning cycle  $t \in [1, T]$  is executed between Lines 4–10. In lines 5–8, each agent  $k \in [1, K]$  is executed. In Line 5, given the ESH,  $B_{GEN}$  is executed to generate the output chunk  $CH_{OG} \in M_{G(k)}$ , using the chunks in  $M_{A(k)}$  and  $M_S$ . In Line 6,  $B_{SUB(k)}$  is applied to submit  $CH_U \in M_{A(k)} \cup M_{G(k)}$  into the buffer  $M_{BA(k)}$  of the agent and into the buffer  $M_{BS}$  to form a chunk set  $\$CH_U$ , based on SPEC-MP. In Line 7,  $B_{UA(k)}$  is executed to update  $CH_M \in M_{A(k)}$  using the chunks stored in the buffer  $M_{BA(k)}$ , based on SPEC-MP. In Line 8, the chunk contained in  $M_{G(k)}$  is processed by facilitator to obtain the best-so-far solution  $\pi_{gb}^{(t)}$ . In Line 10,  $B_{US}$  is executed to update  $M_S$  using the chunks collected in the buffer  $M_{BS}$ , based on SPEC-MP. Finally,  $\pi_{gb}^{(T)}$  is returned and the framework is terminated.

**Algorithm 2.** The execution process of the essential CGO framework

```

Require: The optimization problem  $F_P$  // Other inputs: SPEC-MP, ESH, and a toolbox of chunk and rule instances
1: Facilitator:  $F_P \rightarrow F_R = \langle S, R_M, AUX \rangle$  // Search landscape used by the behaviors of all entities
2:  $B_{INI}$ : SPEC-MP  $\rightarrow \{M_{A(k)} | k \in [1, K]\}, M_S$  // Initialize memories at  $t=0$ 
3: for  $t = 1$  to  $T$  do
4:   for  $k = 1$  to  $K$  do
5:      $B_{GEN(k)}$ :  $\langle \text{ESH}, M_{A(k)}, M_S \rangle \rightarrow CH_{OG} \in M_{G(k)}$  // Agent  $k$ : Generate a new chunk
6:      $B_{SUB(k)}$ :  $\langle \text{SPEC-MP}, M_{A(k)}, CH_{OG} \rangle \rightarrow \langle M_{BA(k)}, M_{BS} \rangle$  // Agent  $k$ : Submit chunk(s) to  $M_{BA(k)}$  &  $M_{BS}$ 
7:      $B_{UA(k)}$ :  $\langle \text{SPEC-MP}, M_{A(k)}^{(t)}, M_{BA(k)} \rangle \rightarrow M_{A(k)}^{(t+1)}$  // Agent  $k$ : Update  $M_A$ 
8:     Facilitator:  $\pi_{gb}^{(t)} \leftarrow CH_{OG}$  if  $R_M(CH_{OG}, \pi_{gb}^{(t)}) \equiv \text{TRUE}$  // Update  $\pi_{gb}^{(t)}$ 
9:   end for
10:   $B_{US}$ :  $\langle \text{SPEC-MP}, M_S^{(t)}, M_{BS} \rangle \rightarrow M_S^{(t+1)}$  // IC: Update  $M_S$  (All buffer memories are cleared at the end of cycle)
11: end for
12: return  $\pi_{gb}^{(T)}$  // Return the best-so-far solution

```

Each row in SPEC-MP defines an updatable relation from  $CH_U$  to  $CH_M$ . The validity of all updatable relations can be easily checked with an *updatable graph* [54] which uses the chunks in  $\{M_A, M_S, M_G\}$  as its nodes, and use updatable relations as its directed arcs. Since the chunks in  $M_G$  are generated in learning cycles, each chunk in  $M_A$  and  $M_S$  can be updated only if this chunk has a directed path originating from a chunk in  $M_G$ .

Each ESH is defined as  $\langle R_{GE}, E_{IG}, CH_{OG} \rangle$ .  $R_{GE}$  is an *elemental generating* rule.  $E_{IG}$  is an ordered list of chunks, where each chunk belongs to  $M_A$  or  $M_S$ .  $CH_{OG}$  is a chunk in  $M_G$ . The  $R_{GE}$  rule takes  $E_{IG}$  as its input, and outputs  $CH_{OG}$  to  $M_G$ .

The chunks in  $M_A$ ,  $M_S$ , and  $M_G$  are of some primary chunk interfaces, and there are three primary rule interfaces, i.e.,  $R_{IE}$ ,  $R_{UE}$ , and  $R_{GE}$  rules. Knowledge components of these primary chunk and rule interfaces could be implemented in the *toolbox* of CGO, and each instance could be called symbolically using its identifier and setting parameters.

Notice that, different optimization algorithms, from simple to complex, can be easily implemented at the symbolic

Further details on the execution of the major behaviors, i.e.,  $B_{INI}$ ,  $B_{GEN}$ ,  $B_{SUB}$ ,  $B_{UA}$  and  $B_{US}$ , are respectively provided in Algorithms 3–7. Notice that only  $B_{GEN}$  is driven by ESH =  $\langle R_{GE}, E_{IG}, CH_{OG} \rangle$ , which uses the  $R_{GE}$  rule to generate a new chunk  $CH_{OG} \in M_G$ , see Algorithm 4. All the other behaviors are driven by SPEC-MP, which maintain  $M_A$  of the agents and  $M_S$  of the IC, see Algorithms 3, 5–7. Each row of SPEC-MP contains  $\langle ID_M, CH_M, R_{IE}, R_{UE}, CH_U \rangle$ . For convenience, SPEC-MP is split into two subtables, SPEC-MP<sub>A</sub> and SPEC-MP<sub>S</sub>, where their  $ID_M$  are respectively  $M_A$  and  $M_S$ . The elements in the  $m$ th row of SPEC-MP<sub>A</sub> and the  $n$ th row of SPEC-MP<sub>S</sub> are respectively indexed by subscripts  $(A, m)$  and  $(S, n)$ . The chunks  $M_{A(m)}$  and  $M_{S(n)}$ , i.e., the  $m$ th chunk in  $M_A$  and the  $n$ th chunk in  $M_S$ , are respectively defined by  $CH_{M(A,m)}$  and  $CH_{M(S,n)}$ . For each row of SPEC-MP, the  $B_{INI}$  behavior uses the  $R_{IE}$  rule to initialize the chunk  $CH_M$ , see Algorithm 3. For each new chunk in  $M_G$ ,  $B_{SUB}$  allocates each  $CH_U$  into the buffers, see Algorithm 5. Then  $B_{UA}$  of each agent and  $B_{US}$  of the IC use each  $R_{UE}$  rule to update each chunk in the memories, see Algorithms 6 and 7.

**Algorithm 3.** Execution of the *initializing* behavior  $B_{INI}$ 

**Require:**  $\langle \text{SPEC-MP} = \langle \text{SPEC-MP}_A, \text{SPEC-MP}_S \rangle$  // Each row of SPEC-MP is  $\langle ID_M, CH_M, R_{IE}, R_{UE}, CH_U \rangle$   
1: **for**  $m = 1$  to  $|\text{SPEC-MP}_A|$  **do** // For each row of SPEC-MP<sub>A</sub>, the subtable of SPEC-MP with  $ID_M = M_A$   
2:  $\{CH_{M(k)} | k \in [1, K]\} = R_{IE(A,m)}()$  // Generate a chunk set of the type  $CH_M$  with  $K$  elements  
3: **for**  $k = 1$  to  $K$  **do**  $CH_{M(k)} \rightarrow M_{A(k,m)}$  **end for** // Initialize  $M_A$  of all agents using the chunks in the set  
4: **end for**  
5: **for**  $n = 1$  to  $|\text{SPEC-MP}_S|$  **do** // For each row of SPEC-MP<sub>S</sub>, the subtable of SPEC-MP with  $ID_M = M_S$   
6:  $CH_M = R_{IE(S,n)}(); CH_M \rightarrow M_{S(n)}$  // Generate a chunk of the type  $CH_M$ ; and initialize  $M_S$  of IC using the chunk  
7: **end for**  
8: **return**  $\{M_{A(k)} | k \in [1, K]\}, M_S$  // The  $M_{A(k)}$  of all agents and  $M_S$  in IC are initialized

**Algorithm 4.** Execution of the *generating* behavior  $B_{GEN}$  of the agent  $k$ 

**Require:**  $\langle \text{ESH}, M_{A(k)}, M_S \rangle$  //  $\text{ESH} = \langle R_{GE}, E_{IG}, CH_{OG} \rangle$   
1:  $\langle M_{A(k)}, M_S \rangle \rightarrow E_{IG}$  // Collect the ordered chunk list  $E_{IG}$  using the chunks from  $M_{A(k)} \cup M_S$   
2:  $CH_{OG} = R_{GE}(E_{IG})$  // Generate new chunk  $CH_{OG}$  by the instance of  $G_{GE}$  rule, using  $E_{IG}$  as the input  
3:  $CH_{OG} \rightarrow M_{G(k)}$  // Store the newly generated chunk  $CH_{OG}$  into  $M_{G(k)}$

**Algorithm 5.** Execution of the *submitting* behavior  $B_{SUB}$  of the agent  $k$ 

**Require:**  $\langle \text{SPEC-MP}, M_{A(k)}, CH_{OG} \rangle$  // Each row of SPEC-MP is  $\langle ID_M, CH_M, R_{IE}, R_{UE}, CH_U \rangle$   
1: **for**  $m = 1$  to  $|\text{SPEC-MP}_A|$  **do** // For each row of SPEC-MP<sub>A</sub>, the subtable of SPEC-MP with  $ID_M = M_A$   
2:  $\langle M_{A(k)}, CH_{OG} \rangle \rightarrow CH_{U(A,m)}$  // Retrieve the chunk  $CH_{U(A,m)}$  from  $M_{A(k)} \cup CH_{OG}$   
3:  $M_{BA(k,m)} = CH_{U(A,m)}$  // Put  $CH_{U(A,m)}$  as the  $m$ th chunk of  $M_{BA(k)}$   
4: **end for**  
5: **for**  $n = 1$  to  $|\text{SPEC-MP}_S|$  **do** // For each row of SPEC-MP<sub>S</sub>, the subtable of SPEC-MP with  $ID_M = M_S$   
6:  $\langle M_{A(k)}, CH_{OG} \rangle \rightarrow CH_{U(S,n)}$  // Retrieve the chunk  $CH_{U(S,n)}$  from  $M_{A(k)} \cup CH_{OG}$   
7:  $M_{BS(n)} = CH_{U(S,n)} \cup M_{BS(n)}$  // Collect  $CH_{U(S,n)}$  into the  $n$ th cell of  $M_{BS}$   
8: **end for**  
9: **return**  $M_{BA(k)}, M_{BS}$  //  $M_{BA(k)}$  is filled by the agent  $k$ , whereas  $M_{BS}$  is filled by all the agents

**Algorithm 6.** Execution of the  $M_A$ -*updating* behavior  $B_{UA}$  of the agent  $k$ 

**Require:**  $\langle \text{SPEC-MP}_A, M_{A(k)}^{(t)}, M_{BA(k)} \rangle$  // Each row of SPEC-MP is  $\langle ID_M, CH_M, R_{IE}, R_{UE}, CH_U \rangle$   
1: **for**  $m = 1$  to  $|\text{SPEC-MP}_A|$  **do** // For each row of SPEC-MP<sub>A</sub>, the subtable of SPEC-MP with  $ID_M = M_A$   
2:  $M_{A(k,m)}^{(t+1)} = R_{UE(A,m)}(M_{A(k,m)}^{(t)}, M_{BA(k,m)})$  // Update the  $m$ th chunk in  $M_{A(k)}$  using the  $m$ th chunk in  $M_{BA(k)}$   
3: **end for**  
4: **return**  $M_{A(k)}^{(t+1)}$  //  $M_{A(k)}^{(t)}$  is updated into  $M_{A(k)}^{(t+1)}$

**Algorithm 7.** Execution of the  $M_S$ -*updating* behavior  $B_{US}$  of IC

**Require:**  $\langle \text{SPEC-MP}_S, M_S^{(t)}, M_{BS} \rangle$  // Each row of SPEC-MP is  $\langle ID_M, CH_M, R_{IE}, R_{UE}, CH_U \rangle$   
1: **for**  $n = 1$  to  $|\text{SPEC-MP}_S|$  **do** // For each row of SPEC-MP<sub>S</sub>, the subtable of SPEC-MP with  $ID_M = M_S$   
2:  $M_{S(n)}^{(t+1)} = R_{UE(S,n)}(M_{S(n)}^{(t)}, M_{BS(n)})$  // Update the  $n$ th chunk in  $M_S$  using the  $n$ th chunk in  $M_{BS}$   
3: **end for**  
4: **return**  $M_S^{(t+1)}$  //  $M_S^{(t)}$  is updated into  $M_S^{(t+1)}$

**5. CGO with ants for solving the TSP**

We will realize both MMAS and CGO-AS in the framework of CGO. The fulfillment can provide us an easy way to identify the similarity and the difference between the two ant systems.

For the TSP, a simple static search landscape  $F_R = \langle S, R_M, AUX \rangle$  can be easily defined. Each possible tour  $\pi$  is a natural state in the problem space  $S$ . The  $R_M$  rule can be realized using the tour length  $f(\pi)$ :  $\forall \pi_{(a)}, \pi_{(b)} \in S, R_M(\pi_{(a)}, \pi_{(b)})$  returns TRUE if and only if  $f(\pi_{(a)}) < f(\pi_{(b)})$ .  $AUX$  simply contains the cost matrix  $D = (d_{ij})$ .

**5.1. Chunk and rule types**

To implement the algorithms in the framework of CGO, we first define a toolbox of knowledge elements of primary chunk and rule types. We only need a few primary chunk types, including a tour  $\pi$ , a tour set  $\$ \pi$ , and a pheromone matrix  $\Psi = (\tau_{ij})$ .

The following  $R_{IE}$  rules are defined. The *randomized*  $R_{IE}$  rule ( $R_{IE}^{RND}$ ) outputs a tour set  $\$ \pi$ , where each element is randomly

generated in  $S$ . The *pheromone matrix*  $R_{IE}$  rule ( $R_{IE}^{PM}$ ) outputs a pheromone matrix with  $\tau_{ij} = \tau_{\max}^{(0)}, \forall i, j$ .

The following  $R_{UE}$  rules are defined. The *greedy*  $R_{UE}$  rule, i.e.,  $R_{UE}^G(\pi_{(M)}, \pi_{(U)}, \pi_{(M)})$  is replaced by  $\pi_{(U)}$  if and only if  $R_M(\pi_{(U)}, \pi_{(M)}) \equiv \text{TRUE}$ . The *pheromone matrix*  $R_{UE}$  rule, i.e.,  $R_{UE}^{PM}(\Psi_{(M)}, \pi_{(U)})$ , updates each  $\tau_{ij} \in \Psi_{(M)}$  by Eq. (2), using the tours in  $\pi_{(U)}$ .

The following  $R_{GE}$  rules are defined. The *social-only*  $R_{GE}$  rule ( $R_{GE}^S$ ) takes a pheromone matrix  $\Psi_{(O)}$  as the input, and constructs one tour  $\pi_{(C)}$  according to Algorithm 1.

The *mixed*  $R_{GE}$  rule ( $R_{GE}^M$ ) takes the chunk list  $\{\pi_{(P)}, \Psi_{(O)}\}$  as the input, and outputs one tour  $\pi_{(C)}$  using Algorithm 8. Algorithm 8 has three parameters,  $p_{ind} \in [0, 1], \sigma_c \geq 0$ , and  $w \in [0, 1]$ . The proportion  $p_c$  is sampled around  $p_{ind}$  using a truncated normal distribution (TND) (Line 3), where the underlying normal distribution has  $\mu = 0$  and  $\sigma = \sigma_c$  and lies within the interval  $[-w, w]$ . In Line 2,  $w$  is examined to ensure that  $p_c \in [0, 1]$ . In Line 4, we define a segment of  $\pi_{(P)}$  with  $N_P$  edges starting from the  $l$ th node, where  $N_P$  is the nearest integer of  $(p_c \cdot N)$ , and  $l$  is the index of city  $i$  in  $\pi_{(P)}$ . The segment is selected into  $\pi_{(C)}$  directly (Lines 6–7). Each remaining city is obtained using  $p_{ij}^{(t)}$  (Line 9), the same as Line 3 in Algorithm 1. Notice that the expectation of  $p_c$  is  $\mathbb{E}(p_c) = p_{ind}$ . Thus the parameter  $p_{ind}$  controls the proportion of input information used in  $\pi_{(P)}$  and  $\Psi_{(O)}$ . If  $p_{ind} = 0$ , Algorithm 8 does not use  $\pi_{(P)}$ , and is equivalent to Algorithm 1; If  $p_{ind} = 1$ , there is  $\pi_{(C)} = \pi_{(P)}$ . Both  $\sigma_c$  and  $w$  are fixed as 0.1 in this paper.

**Algorithm 8.** Construct a tour by an ant (agent) using the mixed information  $\{\pi_{(P)}, \Psi_{(O)}\}$

```

Require:  $\{\pi_{(P)}, \Psi_{(O)}\}$  // The cost matrix  $D$  is a default input, and the parameters include  $p_{ind} \in [0, 1], \sigma_c \geq 0$ , and  $w \in [0, 1]$ 
1:  $\mathcal{N} = \{1, \dots, N\}; i = \text{RND}(\mathcal{N}); \mathcal{N}_i = \mathcal{N} \setminus \{i\}; \pi_{[1]} = i$  // Start from a randomly selected city
2: if  $p_{ind} - w \leq 0$  then  $w = p_{ind}$ ; if  $p_{ind} + w \geq 1$  then  $w = 1 - p_{ind}$  // Preprocess  $w$  to ensure a valid value
3:  $p_c = p_{ind} + \text{SampleTND}(0, \sigma_c, -w, w)$  // Sample from the truncated normal distribution (TND) within  $(-w, w)$ 
4:  $N_P = \text{Round}(p_c \cdot N); l = \text{GetIndex}(i, \pi_{(P)})$  // Define a segment of  $\pi_{(P)}$  with  $N_P$  edges, starting from the  $l$ th node
5: for  $n = 2$  to  $N$  do
6:   if  $n \leq N_P + 1$  then
7:      $j = \pi_{(P)}[l]; l = l + 1$ ; if  $l > N$  then  $l = 1$  // Select the  $l$ th node of  $\pi_{(P)}$  and go to the next node of  $\pi_{(P)}$ 
8:   else
9:     Select the next city  $j \in \mathcal{N}_i$  with the probability  $p_{ij}^{(t)}$ , using Eq. 1 with  $\Psi_{(O)}$  and  $D$ 
10:  end if
11:   $\pi_{[n]} = j; \mathcal{N}_j = \mathcal{N}_i \setminus \{j\}; i = j$  // Move to the next nodes  $j$ 
12: end for
13: return  $\pi_{(C)} = (\pi_{[1]}, \dots, \pi_{[N]})$  // Return the new tour
    
```

The *3-opt*  $R_{GE}$  rule ( $R_{GE}^{3O}$ ) is the same as the 3-opt local search algorithm used in [48]. The 3-opt local search algorithm proceeds by systematically testing the incumbent tour  $\pi_{(C)}$ , with some standard speed-up techniques using nearest neighbors [3,48], and with the technique of *don't look bits* on each node [3]. Notice that the tour  $\pi_{(C)}$  can be improved by replacing at most three edges in each test. By default, the candidate list at length 20 is considered [48].

The macro  $R_{GE}$  rule  $R_{GE}^{S+3O}$  is defined as the tuple  $(R_{GE}^S, R_{GE}^{3O})$ , where the input is  $\Psi_{(O)}$ , and the output of  $R_{GE}^S$  is further processed by  $R_{GE}^{3O}$ .

The macro  $R_{GE}$  rule  $R_{GE}^{M+3O}$  is defined as the tuple  $(R_{GE}^M, R_{GE}^{3O})$ , where the input is  $\{\pi_{(P)}, \Psi_{(O)}\}$ , and the output of  $R_{GE}^M$  further processed by  $R_{GE}^{3O}$ .

### 5.2. SPEC-MP and ESHs

For MMAS, the memory elements are defined as follows:  $M_A$  is empty,  $M_S$  contains the pheromone matrix  $\Psi$ , and  $M_G$  contains one tour  $\pi_C$ . For CGO-AS, the memory elements are defined as follows:  $M_A$  contains one tour  $\pi_P$ ,  $M_S$  contains the pheromone matrix  $\Psi$ , and  $M_G$  contains one tour  $\pi_C$ .

**Table 1**  
The memory protocol rows in SPEC-MP for CGO-AS.

$ID_M$	$CH_M$	$R_{IE}$ instance	$R_{UE}$ instance	$CH_U$
$M_A$	$\pi_P$	$R_{IE}^{RND}$	$R_{UE}^G$	$\pi_C$
$M_S$	$\Psi$	$R_{IE}^{PM}$	$R_{UE}^{PM}$	$\pi_C$

**Table 2**  
The list of embedded search heuristics (ESHs).

ESH	$R_{GE}$ instance	$E_{IG}$	$CH_{OG}$
MMAS	$R_{GE}^S$	$\{\Psi\}$	$\pi_C$
MMAS <sub>3opt</sub>	$R_{GE}^{S+3O}$	$\{\Psi\}$	$\pi_C$
CGO-AS	$R_{GE}^M$	$\{\pi_P, \Psi\}$	$\pi_C$
CGO-AS <sub>3opt</sub>	$R_{GE}^{M+3O}$	$\{\pi_P, \Psi\}$	$\pi_C$

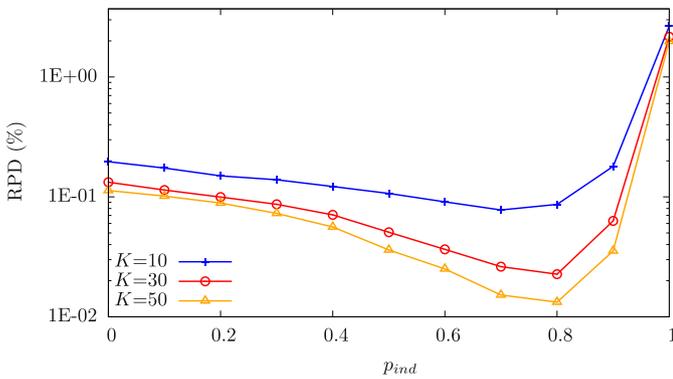
Both MMAS and CGO-AS can work on the same SPEC-MP, as defined in Table 1. The difference between them is that MMAS only uses  $\Psi$  in  $M_S$ , whereas CGO-AS also uses  $\pi_P$  in  $M_A$ . Normally, some additional knowledge might be embedded with the SPEC-MP. For example, as  $R_{UE}^G$  is used,  $\pi_P$  in  $M_A$  always retains the personal best solution for an agent.

Table 2 lists the embedded search heuristics (ESHs) of this work, including MMAS, MMAS<sub>3opt</sub>, CGO-AS, and CGO-AS<sub>3opt</sub>, where MMAS<sub>3opt</sub> and CGO-AS<sub>3opt</sub> also apply the 3-opt local search. Note that the  $R_{GE}^M$  rules in CGO-AS and CGO-AS<sub>3opt</sub> both have a setting parameter  $p_{ind} \in [0, 1]$  in Algorithm 8, and are respectively reduced to MMAS and MMAS<sub>3opt</sub> if  $p_{ind} = 0$ .

### 5.3. Brief summary

Although CGO looks a little bit complex, the realization of algorithms in this framework is smooth. CGO-AS can be reduced into an implementation of algorithmic components (chunks and rules in Section 5.1) with a simple script description (SPEC-MP and ESHs in Tables 1 and 2).

We can easily identify the similarities and differences between MMAS and CGO-AS, base on the framework of CGO as shown in Fig. 1. Each ant is represented as an agent in CGO. The pheromone matrix is stored in the social memory  $M_S$ , and it is updated using  $M_{BS}$  and  $B_{US}$ . There are two main differences. First, CGO-AS uses additional modules including the individual memory  $M_A$  and its maintenance modules  $M_{BA}$  and  $B_{UA}$ . Second, in  $B_{GEN}$ , the elemental generating rule is changed from Algorithm 1 (in MMAS), which only uses pheromone trails, to Algorithm 8 (in CGO-AS), which uses both individual and social learning with the proportion controlled by  $p_{ind}$ .



**Fig. 2.** Average RPD of mean results by CGO-AS<sub>3opt</sub> with  $K = \{10, 30, 50\}$  and  $p_{ind} \in [0, 1]$ .

With the two differences, the memory form is transformed from one single social memory into one social plus multiple individual memories, and the learning form is upgraded from a pure social learning into a mixed social and individual learning. The framework of CGO provides a support for the self-organized interactions between multiple memories and behaviors.

## 6. Results and discussion

We conduct a series of experiments to evaluate the performance of two algorithms, CGO-AS<sub>3opt</sub> and MMAS<sub>3opt</sub> that incorporates the 3-opt local search rule  $R_{GE}^{3O}$ . Due to an application of the 3-opt and other local search heuristic,<sup>1</sup> local optima of the search landscape of TSP exhibits a “big valley” structure [39,48]. This suggests that the high-quality tours tend to concentrate on a very small subspace around the optimal tour(s). On the basic group parameters, we consider  $T=500$ , and  $K \in \{10, 30, 50\}$ . For the 3-opt local search, 20 nearest neighbors are used. On the other parameters, we consider default settings, including  $\alpha = 1$ ,  $\beta = 2$ , and  $p_{best} = 0.05$ . In MMAS [48], a schedule is used to alternate the pheromone-trail update between  $\pi_{ib}^{(t)}$  and  $\pi_{gb}^{(t)}$ . In the first 25, 75, 125, 250, and the later iterations, the intervals using  $\pi_{gb}^{(t)}$  are respectively 25, 5, 3, 2, and 1.

We perform experiments on the widely-used benchmark instances in TSPLIB [45]. For each problem instance, 100 independent runs have been performed for obtaining the mean results in statistic. For each mean result  $f$ , we report  $100 \cdot (f - \bar{f}) / \bar{f}$ , the relative percentage deviation (RPD) as an evaluation value of  $f$ , and  $100 \cdot \sigma / \bar{f}$ , as an evaluation value of the standard deviation (SD)  $\sigma$ , where  $\bar{f}$  is the optimal value. The minimal RPD of the optimal solution is 0.

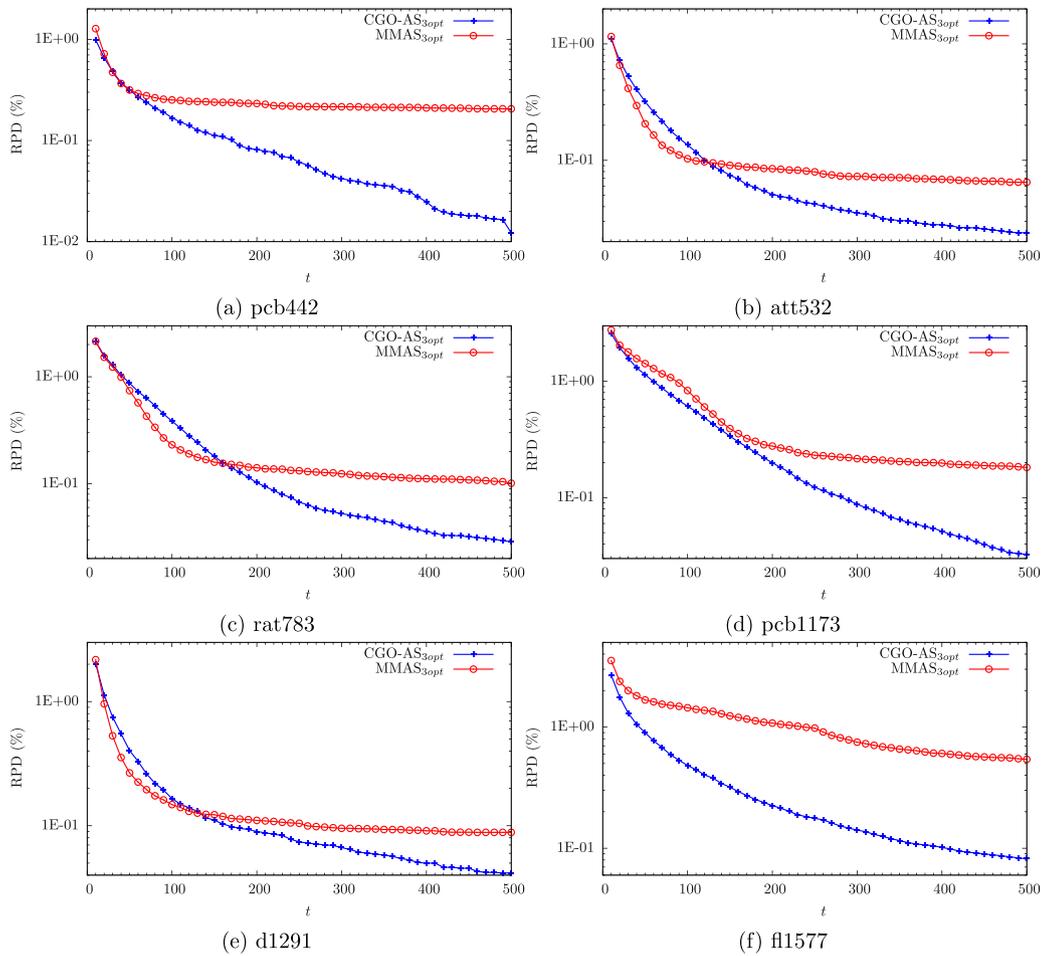
We first evaluate the performance on 10 benchmark instances in the range of  $N$  from 51 to 1577 (as listed in Table A1(a)–(c) of Appendix A), which are a set of instances frequently used in the literature. Fig. 2 shows the average RPD of the mean by CGO-AS<sub>3opt</sub> with  $K = \{10, 30, 50\}$ , for  $p_{ind}$  in the range from 0 to 1. For the agents in CGO-AS,  $p_{ind}$  controls the proportion using knowledge between individual and social memory, i.e. the weighting balance between individual and social learning. As shown in the figure, CGO-AS<sub>3opt</sub> cannot reach a good performance at either end, i.e., in the case of  $p_{ind} = 0$  or  $p_{ind} = 1$ . In the case of  $p_{ind} = 0$ , ants only use social memory (pheromone trails), for which MMAS<sub>3opt</sub> is in fact this special case of CGO-AS<sub>3opt</sub>. In the case of  $p_{ind} = 1$ , ants only use their own

individual (route) memories and perform local searching. Interestingly, the results showed that CGO-AS<sub>3opt</sub> reached a much better performance when ants take a mixed usage combining both individual and social memory together. In this case, ants use two parts of learning. One part of learning uses social memory that contains an accumulated adaptive knowledge [6] for accelerating the learning process. Another part of learning uses individual route memory that preserves novel patterns [42] for supporting the capability to escape from some maladaptive outcomes [6]. As shown in Fig. 2, CGO-AS<sub>3opt</sub> reaches the best performance of searching at around  $p_{ind} = 0.8$ . This means that the best searching relies more on individual memory, which is called as socially biased individual learning (SBIL) in the field of animal learning [22]. The result is interpretative from the viewpoint of searching for optimization. Each ant in CGO-AS performs a local searching based on its individual memory; at the same time, it also efficiently search “big valleys” of TSP landscape with the guide of the high-quality heuristic cues in the social memory accumulated by all ants (simulating the pheromone trails of natural ants). If ants only use individual information, i.e. in the case of  $p_{ind} = 1$ , the search would more likely be trapped to the local minima. In contrast, if ants only use social information (such as the pheromone matrix used by the existing ant systems), i.e. in the case of  $p_{ind} = 0$ , it is challenging to adaptively maintain a diversity in searching. Aiming to prevent the search of AS from a premature convergence, previous research on MMAS [48] had attempted to introduce the mechanism linking diversity into pheromone trail by re-initialization, but showed a very limited success.

More detailed results on the 10 instances are provided in Appendix A. For each tested instance, Table A1(a)–(c) gives the ratios of the runs reaching the optimal solution (Best), the RPD of the mean values (Mean), and the standard deviations (SD) by CGO-AS<sub>3opt</sub> (with  $p_{ind} = 0.8$ ) and MMAS<sub>3opt</sub> for the experiments with  $K \in \{10, 30, 50\}$  respectively. The results show that CGO-AS<sub>3opt</sub> has achieved a significant better performance than MMAS<sub>3opt</sub>. In comparison with MMAS<sub>3opt</sub>, for all the instances, CGO-AS<sub>3opt</sub> gains a much bigger ratio of the runs reaching the optimal solution, a smaller RPD of the mean value, and a lower standard deviation, see Table A1(a)–(c). CGO-AS<sub>3opt</sub> with  $K = 10$  outperforms MMAS<sub>3opt</sub> with  $K = 50$  on the RPD of the mean value and the standard deviation, and beats MMAS<sub>3opt</sub> with  $K = 30$  completely on all the resulting values. In the case with  $K = 50$ , CGO-AS<sub>3opt</sub> is able to solve two more instances (d198 and lin318) than MMAS<sub>3opt</sub> in all runs, and to find the optimal solutions for all instances including fl1577.

To observe more information for understanding the learning process more clearly, we show Fig. 3, the RPD results of CGO-AS<sub>3opt</sub> (with  $p_{ind} = 0.8$ ) and MMAS<sub>3opt</sub> for six larger TSP instances with  $K = 30$  over 500 learning cycles. For MMAS<sub>3opt</sub> (which only uses social memory), its learning process quickly stagnated at the local and lower-quality minima, although it holds a fast learning speed in its early learning cycles. For all the tested instances here, CGO-AS<sub>3opt</sub> outperforms MMAS<sub>3opt</sub> to reach a much better quality in solution. It is interesting that although MMAS<sub>3opt</sub> has a quicker learning speed at the very beginning learning cycles, CGO-AS quickly catches up the learning speed of MMAS<sub>3opt</sub>, and keeps a much faster learning pace than MMAS<sub>3opt</sub> in the following later learning cycles, and finally reaches a better solution. For some tests (such as fl1577), CGO-AS has a faster learning speed than MMAS<sub>3opt</sub> in the whole learning cycles. Fig. 4 gives the comparison in population diversity between CGO-AS<sub>3opt</sub> (with  $p_{ind} = 0.8$ ) and MMAS<sub>3opt</sub>, by showing the relative distances between  $\{\pi_{C(k)}^{(t)} | k \in [1, K]\}$  (i.e., the set of newly generated states by the agents) of CGO-AS<sub>3opt</sub> and MMAS<sub>3opt</sub> for six larger TSP instances with  $K = 30$  over 500 learning cycles. The relative distance is defined as  $\bar{d}/N$ , where  $\bar{d} = \sum_{k_1 < k_2} d(\pi_{C(k_1)}^{(t)}, \pi_{C(k_2)}^{(t)}) / C_2^K$ , and the distance  $d$  between any two states is given by the number of different edges. As shown in Fig. 4,

<sup>1</sup> Note that the LK heuristic [36] and its variants [2,30] are often considered as the best performing local search strategies, with respect to the solution quality. Nevertheless, the LK heuristic is more difficult to implement than 2-opt and 3-opt, and it requires careful fine-tuning to run efficiently [48].



**Fig. 3.** RPD of mean results by CGO-AS<sub>3opt</sub> and MMAS<sub>3opt</sub> with  $K = 30$  over the 500 learning cycles.

CGO-AS<sub>3opt</sub> holds a higher diversity in population than MMAS<sub>3opt</sub> over the learning cycles for almost all the test instances. The population diversity among the agents in CGO-AS<sub>3opt</sub> is adaptively maintained by their individual memory in the learning process. Each agent maintains the personal best state  $\pi_p$  in its  $M_A$  (based on the specification in Table 1), and generates each new state largely inheriting from the high-quality information in its  $M_A$ , as  $p_{ind}$  in Algorithm 8 is sufficiently large. Holding population diversity is important for optimization, as it has been shown to play a significant role of effectiveness in the problem solving of human groups [44,34].

Next, we evaluate the performance of CGO-AS<sub>3opt</sub> by comparing it with other ant systems and some recently published algorithms,

in terms of the RPD of mean results and standard deviations in optimization respective to the best solutions. We run CGO-AS<sub>3opt</sub> on AMD Phenom II 3.4 GHz, and report the computational speed of CGO-AS<sub>3opt</sub> with the CPU time (in seconds). In Tables 3 and 4, the symbol “–” means that no result was provided in the references.

Table 3 gives the comparison among CGO-AS<sub>3opt</sub> with  $K=10$  and two ant systems, PSO-ACO-3Opt [38] and FOGS-ACO [46]. PSO-ACO-3Opt is an ACO algorithm with a set of performance parameters that are optimized using both PSO and the 3-opt local search operator. FOGS-ACO is a hybrid algorithm of the Fast Opposite Gradient Search (FOGS) and ACO. The test is performed on a set of TSPLIB instances, where the number of nodes is from 51 to 200 used by the two ant systems. As shown in Table 3, for all the

**Table 3**  
Results by CGO-AS<sub>3opt</sub> with  $K=10$ , PSO-ACO-3Opt, and FOGS-ACO.

Instance	$f_{opt}$	CGO-AS <sub>3opt</sub>			PSO-ACO-3opt		FOGS-ACO	
		Mean (%)	SD (%)	Time	Mean (%)	SD (%)	Mean (%)	SD (%)
eil51	426	0.000	0.0000	0.002	0.106	0.1432	2.406	1.2465
berlin52	7542	0.000	0.0000	0.001	0.016	0.0314	0.526	0.6310
st70	675	0.000	0.0000	0.003	0.474	0.2178	2.873	1.2356
eil76	538	0.000	0.0000	0.005	0.056	0.0874	1.976	1.0762
pr76	108159	0.000	0.0000	0.002	–	–	2.522	1.8274
rat99	1211	0.000	0.0000	0.013	1.354	0.1635	–	–
kroA100	21282	0.000	0.0000	0.003	0.766	0.3676	0.682	2.9804
rd100	7910	0.000	0.0000	0.005	–	–	2.248	1.1876
eil101	629	0.000	0.0000	0.014	0.588	0.3370	3.919	3.4118
lin105	14379	0.000	0.0000	0.002	0.001	0.0033	–	–
ch150	6528	0.000	0.0000	0.029	0.551	0.4225	–	–
kroA200	29368	0.000	0.0000	0.057	0.947	0.3906	7.314	2.2953

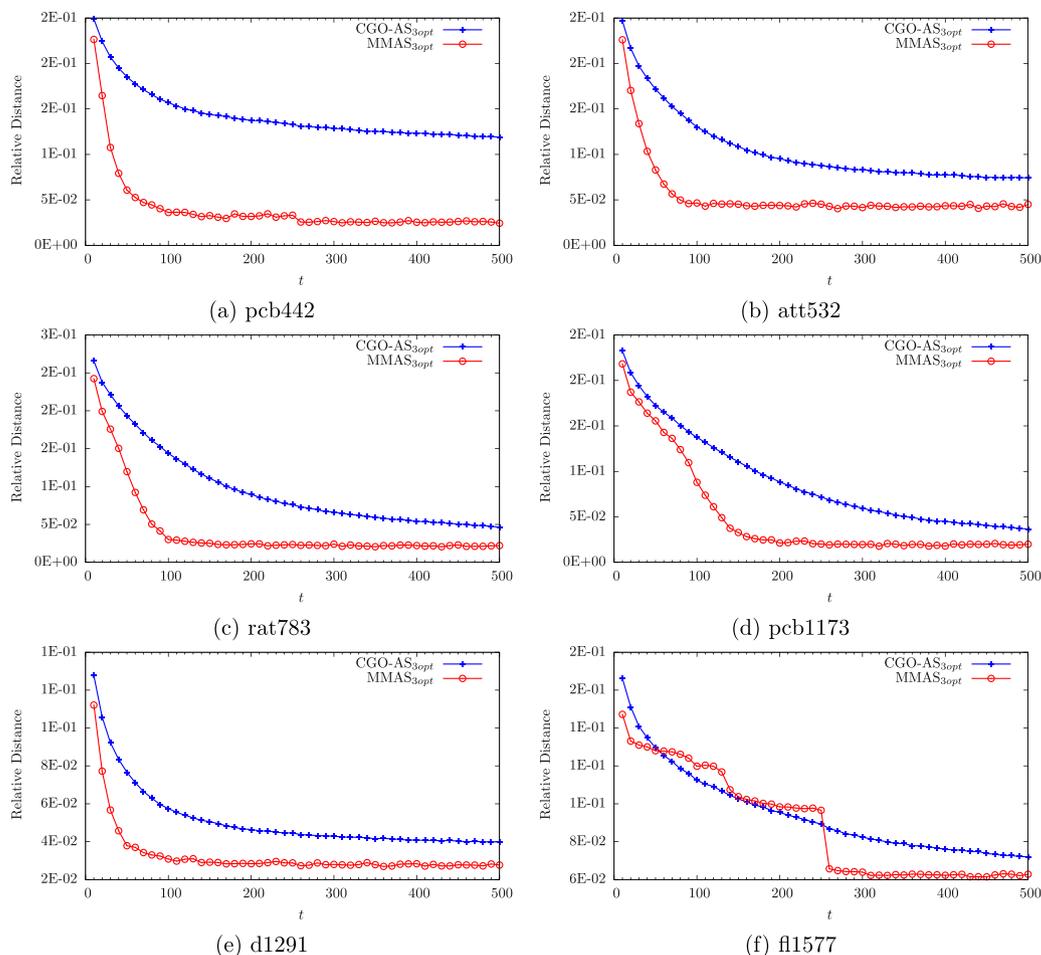


Fig. 4. Population diversity by CGO-AS<sub>3opt</sub> and MMAS<sub>3opt</sub> with  $K=30$  over the 500 learning cycles.

Table 4

Results by CGO-AS<sub>3opt</sub> with  $K=50$ , DIWO, and DCS.

Instance	$f_{opt}$	CGO-AS <sub>3opt</sub>			DIWO		DCS	
		Mean (%)	SD (%)	Time	Mean (%)	SD (%)	Mean (%)	SD (%)
tsp225	3916	0.000	0.000	0.16	2.395	2.821	1.092	0.529
pr226	80369	0.000	0.000	0.07	0.224	0.192	0.022	0.075
pr264	49135	0.000	0.000	0.05	–	–	0.249	0.326
a280	2579	0.000	0.000	0.07	0.768	0.700	0.517	0.460
pr299	48191	0.000	0.000	0.20	–	–	0.580	0.273
lin318	42029	0.000	0.000	0.84	–	–	0.965	0.441
rd400	15281	0.000	0.001	1.81	2.423	0.863	1.654	0.396
fl417	11861	0.000	0.000	0.26	–	–	0.418	0.172
pcb442	50778	0.005	0.027	1.81	2.173	0.799	–	–
pr439	107217	0.000	0.003	1.23	–	–	0.693	0.409
att532	27686	0.013	0.021	4.60	1.874	0.802	–	–
rat575	6773	0.023	0.019	4.46	–	–	2.713	0.528
rat783	8806	0.012	0.017	8.74	–	–	3.444	0.433
pr1002	259045	0.028	0.034	12.81	3.187	2.604	3.700	0.435

instances, CGO-AS<sub>3opt</sub> reaches the optimal value in all runs, and outperforms both PSO-ACO-30pt and FOGS-ACO.

Table 4 gives the comparison between CGO-AS<sub>3opt</sub> with  $K=50$  and two other recently published optimization algorithms, DIWO [55] and DCS [43]. DIWO is a discrete invasive weed optimization (IWO) algorithm with the 3-opt local search operator. DCS is a discrete cuckoo search algorithm for solving the TSP, which can reconstructs its population to introduce a new category of cuckoos. The test is performed on a set of TSPLIB instances, where the number of nodes is from 225 to 1002 used by the two optimization

algorithms. As shown in Table 4, for all the instances, CGO-AS<sub>3opt</sub> again outperforms both DIWO and FOGS-ACO. CGO-AS<sub>3opt</sub> reaches the optimal value for seven of the test instances in all runs, and approaches to the near optimal value for the other test instances.

## 7. Conclusion

In this paper, we presented CGO-AS, a generalized ant system that is implemented in the framework of cooperative group

optimization. AS is an algorithm simulating the foraging system that uses pheromone trails in ants colonies. CGO is a framework to support the cooperative search process by a group of agents. CGO-AS has combined and used both individual route memory and social pheromone trails to simulate the intelligence of natural ants, therefore it enables us to leverage the power of mixed individual and social learning. We have not attempted to provide a complete comparison in performance between CGO-AS and the existing algorithms on various problems. Rather, we aim at showing the benefit of using mixed social and individual learning in optimization. We tested the performance of CGO-AS for elucidating the weighting balance between individual and social learning, and compared it with the existing AS systems and some recently published algorithms, using the well-known traveling salesman problem (TSP) as a benchmark.

The results on the instances in TSPLIB showed that the group of agents (ants) with a mixed usage of individual and social memory reaches a much better performance of search than the systems using either individual memory or social memory only. The best performance is gained under the condition when agents use individual memory as their primary information source, and simultaneously also use social memory as their searching guidance.

The tests showed that CGO-AS not only reaches a better quality in solution, but also holds a faster learning speed, especially in later learning cycles. The benefit of optimization may be due to the introduced mechanism in the CGO-AS algorithm that adaptively maintains the population diversity using the information learned and stored in the individual memory of each agent, and also accelerates the learning process using the knowledge accumulated in the social memory of agents. The performance of CGO-AS<sub>3opt</sub> turned out to be competitive in comparison with the existing AS systems and some recent published algorithms, including MMAS<sub>3opt</sub>, PSO-ACO-3Opt, FOGS-ACO, DIWO, and DCS.

**Appendix A.**

Table A1(a)–(c) gives the ratios of the runs reaching the optimal solution (Best), the RPD of the mean values (Mean), and the standard deviations (SD) by CGO-AS<sub>3opt</sub> (with  $p_{ind} = 0.8$ ) and MMAS<sub>3opt</sub> for the experiments with  $K \in \{10, 30, 50\}$  respectively, on 10 benchmark instances in the range of  $N$  from 51 to 1577, which are a set of instances frequently used in the literature.

**Table A1**  
Results by CGO-AS<sub>3opt</sub> and MMAS<sub>3opt</sub> on 10 benchmark instances in the range of  $N$  from 51 to 1577.

Instance	$f_{opt}$	CGO-AS <sub>3opt</sub>			MMAS <sub>3opt</sub>		
		Best (%)	Mean (%)	SD (%)	Best (%)	Mean (%)	SD (%)
<i>(a) K = 10</i>							
eil51	426	100	0.000	0.000	100	0.000	0.000
kroA100	21282	100	0.000	0.000	100	0.000	0.000
d198	15780	70	0.002	0.004	35	0.007	0.013
lin318	42029	59	0.056	0.086	35	0.185	0.151
pcb442	50778	14	0.128	0.119	3	0.260	0.121
att532	27686	9	0.069	0.043	5	0.091	0.055
rat783	8806	1	0.114	0.069	3	0.187	0.119
pcb1173	56892	1	0.185	0.128	0	0.338	0.185
d1291	50801	9	0.115	0.082	3	0.136	0.113
fl1577	22249	0	0.197	0.132	0	0.762	0.252
Average		36.3	0.087	0.087	28.4	0.197	0.125
<i>(b) K = 30</i>							
eil51	426	100	0.000	0.000	100	0.000	0.000
kroA100	21282	100	0.000	0.000	100	0.000	0.000
d198	15780	97	0.000	0.001	44	0.003	0.003
lin318	42029	96	0.006	0.027	51	0.139	0.149
pcb442	50778	70	0.012	0.038	10	0.206	0.121
att532	27686	42	0.024	0.026	17	0.065	0.046
rat783	8806	33	0.029	0.034	15	0.101	0.078
pcb1173	56892	9	0.032	0.038	2	0.182	0.144
d1291	50801	21	0.042	0.038	11	0.088	0.078
fl1577	22249	0	0.082	0.045	0	0.541	0.183
Average		56.8	0.023	0.031	35.0	0.133	0.103
<i>(c) K = 50</i>							
eil51	426	100	0.000	0.000	100	0.000	0.000
kroA100	21282	100	0.000	0.000	100	0.000	0.000
d198	15780	100	0.000	0.000	53	0.003	0.003
lin318	42029	100	0.000	0.000	68	0.091	0.134
pcb442	50778	86	0.005	0.027	20	0.187	0.132
att532	27686	66	0.013	0.021	25	0.064	0.056
rat783	8806	56	0.012	0.017	26	0.078	0.076
pcb1173	56892	30	0.014	0.023	4	0.126	0.123
d1291	50801	31	0.031	0.030	13	0.087	0.080
fl1577	22249	1	0.057	0.028	0	0.494	0.202
Average		67.0	0.013	0.015	40.9	0.113	0.102

## References

- [1] J.R. Anderson, Human symbol manipulation within an integrated cognitive architecture, *Cognit. Sci.* 29 (3) (2005) 313–341.
- [2] D. Applegate, W. Cook, A. Rohe, Chained Lin-Kernighan for large traveling salesman problems, *INFORMS J. Comput.* 15 (1) (2003) 82–92.
- [3] J.J. Bentley, Fast algorithms for geometric traveling salesman problems, *ORSA J. Comput.* 4 (4) (1992) 387–411.
- [4] S. Bolek, H. Wolf, Food searches and guiding structures in North African desert ants, *cataglyphis*, *J. Comp. Physiol. A* 201 (6) (2015) 631–644.
- [5] E. Bonabeau, M. Dorigo, G. Theraulaz, Inspiration for optimization from social insect behaviour, *Nature* 406 (2000) 39–42.
- [6] R. Boyd, P.J. Richerson, J. Henrich, The cultural niche: why social learning is essential for human adaptation, *Proc. Natl. Acad. Sci.* 108 (2011) 10918–10925.
- [7] B. Bullnheimer, R.F. Hartl, C. Strauss, An improved ant system algorithm for the vehicle routing problem, *Ann. Oper. Res.* 89 (1999) 319–328.
- [8] K. Cheng, P. Schultheiss, S. Schwarz, A. Wystrach, R. Wehner, Beginnings of a synthetic approach to desert ant navigation, *Behav. Processes* 102 (2014) 51–61.
- [9] M. Collett, How navigational guidance systems are combined in a desert ant, *Curr. Biol.* 22 (10) (2012) 927–932.
- [10] M. Collett, A desert ant's memory of recent visual experience and the control of route guidance, *Proc. R. Soc. Lond. B: Biol. Sci.* 281 (1787) (2014) 20140634.
- [11] M. Collett, T.S. Collett, S. Bisch, R. Wehner, Local and global vectors in desert ant navigation, *Nature* 394 (6690) (1998) 269–272.
- [12] T. Collett, M. Collett, Memory use in insect visual navigation, *Nat. Rev. Neurosci.* 3 (7) (2002) 542–552.
- [13] T. Collett, P. Graham, V. Durier, Route learning by insects, *Cur. Opin. Neurobiol.* 13 (6) (2003) 718–725.
- [14] T.J. Czaczkes, C. Grüter, L. Ellis, E. Wood, F.L. Ratnieks, Ant foraging on complex trails: route learning and the role of trail pheromones in *Lasius niger*, *J. Exp. Biol.* 216 (2) (2013) 188–197.
- [15] T.J. Czaczkes, C. Grüter, F.L.W. Ratnieks, Trail pheromones: an integrative view of their role in social insect colony organization, *Annu. Rev. Entomol.* 60 (2015) 581–599.
- [16] É. Danchin, L.-A. Giraldeau, T. Valone, R. Wagner, Public information: from nosy neighbors to cultural evolution, *Science* 305 (5683) (2004) 487–491.
- [17] J. Deneubourg, S. Aron, S. Goss, J. Pasteels, The self-organizing exploratory pattern of the Argentine ant, *J. Insect Behav.* 3 (2) (1990) 159–168.
- [18] A. Dennis, J. Valacich, Computer brainstorming: more heads are better than one, *J. Appl. Psychol.* 78 (4) (1993) 531–537.
- [19] M. Dorigo, L. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, *IEEE Trans. Evol. Comput.* 1 (1) (1997) 53–66.
- [20] M. Dorigo, V. Maniezzo, A. Colnori, Ant system: optimization by a colony of cooperating agents, *IEEE Trans. Syst. Man Cybern. Part B: Cybern.* 26 (1) (1996) 29–41.
- [21] K.A. Ericsson, W. Kintsch, Long-term working memory, *Psychol. Rev.* 102 (2) (1995) 211–245.
- [22] B.G. Galef, Why behaviour patterns that animals learn socially are locally adaptive, *Anim. Behav.* 49 (5) (1995) 1325–1334.
- [23] M. Giurfa, E. Capaldi, Vectors, routes and maps: new discoveries about navigation in insects, *Trends Neurosci.* 22 (6) (1999) 237–242.
- [24] A.M. Glenberg, What memory is for, *Behav. Brain Sci.* 20 (1) (1997) 1–55.
- [25] J.A. Goncalo, B.M. Staw, Individualism–collectivism and group creativity, *Organ. Behav. Hum. Decis. Process.* 100 (2006) 96–109.
- [26] S. Goss, S. Aron, J. Deneubourg, J. Pasteels, Self-organized shortcuts in the Argentine ant, *Naturwissenschaften* 76 (12) (1989) 579–581.
- [27] C. Grüter, T.J. Czaczkes, F.L. Ratnieks, Decision making in ant foragers (*Lasius niger*) facing conflicting private and social information, *Behav. Ecol. Sociobiol.* 65 (2) (2011) 141–148.
- [28] M. Gunduz, M.S. Kiran, E. Özceylan, A hierarchic approach based on swarm intelligence to solve the traveling salesman problem, *Turk. J. Electr. Eng. Comput. Sci.* 23 (2015) 103–117.
- [29] R. Harris, N. de Ibarra, P. Graham, T. Collett, Ant navigation: priming of visual route memories, *Nature* 438 (7066) (2005) 302.
- [30] K. Helsgaun, An effective implementation of the Lin-Kernighan traveling salesman heuristic, *Eur. J. Oper. Res.* 126 (1) (2000) 106–130.
- [31] B. Hölldobler, E. Wilson, *The Ants*, Harvard University Press (1990).
- [32] D. Jackson, F. Ratnieks, Communication in ants, *Curr. Biol.* 16 (15) (2006) R570–R574.
- [33] S. Kauffman, S. Levin, Towards a general theory of adaptive walks on rugged landscapes, *J. Theor. Biol.* 128 (1) (1987) 11–45.
- [34] S. Kavadias, S.C. Sommer, The effects of problem structure and team diversity on brainstorming effectiveness, *Manag. Sci.* 55 (12) (2009) 1899–1913.
- [35] T. Liao, K. Socha, M. Montes de Oca, T. Stutzle, M. Dorigo, et al., Ant colony optimization for mixed-variable optimization problems, *IEEE Trans. Evol. Comput.* 18 (4) (2014) 503–518.
- [36] S. Lin, B.W. Kernighan, An effective heuristic algorithm for the traveling-salesman problem, *Oper. Res.* 21 (2) (1973) 498–516.
- [37] D. Macquart, L. Garnier, M. Combe, G. Beugnon, Ant navigation en route to the goal: signature routes facilitate way-finding of *gigantiops destructor*, *J. Comp. Physiol. A* 192 (3) (2006) 221–234.
- [38] M. Mahi, Ö.K. Baykan, H. Kodaz, A new hybrid method based on particle swarm optimization, ant colony optimization and 3-opt algorithms for traveling salesman problem, *Appl. Soft Comput.* 30 (2015) 484–490.
- [39] P. Merz, B. Freisleben, Memetic algorithms for the traveling salesman problem, *Complex Syst.* 13 (4) (2001) 297–345.
- [40] E. Morgan, Trail pheromones of ants, *Physiol. Entomol.* 34 (1) (2009) 1–17.
- [41] S. Nallaperuma, M. Wagner, F. Neumann, Analyzing the effects of instance features and algorithm parameters for MAX-MIN ant system and the traveling salesperson problem, *Front. Robot. AI* 2 (2015) 18.
- [42] C.J. Nemeth, Differential contributions of majority and minority influence, *Psychol. Rev.* 93 (1) (1986) 23–32.
- [43] A. Ouaraab, B. Ahiod, X.-S. Yang, Discrete cuckoo search algorithm for the travelling salesman problem, *Neural Comput. Appl.* 24 (7–8) (2014) 1659–1669.
- [44] P.B. Paulus, Groups, teams, and creativity: the creative potential of idea-generating groups, *Appl. Psychol.* 49 (2) (2000) 237–262.
- [45] G. Reinelt, TSPLIB – a traveling salesman problem library, *ORSA J. Comput.* 3 (1991) 376–384.
- [46] T. Saenphon, S. Phimoltares, C. Lursinsap, Combining new fast opposite gradient search with ant colony optimization for solving travelling salesman problem, *Eng. Appl. Artif. Intell.* 35 (2014) 324–334.
- [47] S. Sommer, C. von Beeren, R. Wehner, Multiroute memories in desert ants, *Proc. Natl. Acad. Sci.* 105 (1) (2008) 317–322.
- [48] T. Stützle, H. Hoos, MAX-MIN ant system, *Future Gen. Comput. Syst.* 16 (8) (2000) 889–914.
- [49] M. Tomasello, A. Kruger, H. Ratner, Cultural learning, *Behav. Brain Sci.* 16 (3) (1993) 495–511.
- [50] R. Wehner, Desert ant navigation: how miniature brains solve complex tasks, *J. Comp. Physiol. A* 189 (8) (2003) 579–588.
- [51] R. Wehner, M. Boyer, F. Loertscher, S. Sommer, U. Menzi, Ant navigation: one-way routes rather than maps, *Curr. Biol.* 16 (1) (2006) 75–79.
- [52] H. Wolf, Desert ants adjust their approach to a foraging site according to experience, *Behav. Ecol. Sociobiol.* 62 (3) (2008) 415–425.
- [53] A. Wystrach, M. Mangan, A. Philippides, P. Graham, Snapshots in ants? New interpretations of paradigmatic experiments, *J. Exp. Biol.* 216 (10) (2013) 1766–1770.
- [54] X.-F. Xie, J. Liu, Z.-J. Wang, A cooperative group optimization system, *Soft Comput.* 18 (3) (2014) 469–495.
- [55] Y. Zhou, Q. Luo, H. Chen, A. He, J. Wu, A discrete invasive weed optimization algorithm for solving traveling salesman problem, *Neurocomputing* 151 (2015) 1227–1236.