Front.Comput.Sci.
DOI

RESEARCH ARTICLE

On the Analysis of Ant Colony Optimization for the Maximum Independent Set Problem (Extended Version)

Xiaoyun XIA (✉)[1], Xue PENG [2], Weizhi LIAO [1]

1 College of Information Science and Engineering, Jiaxing University, Jiaxing 314001, China
2 School of Mathematics and Systems Science, Guangdong Polytechnic Normal University, Guangzhou 510665, China

Abstract Ant Colony Optimization (ACO) is a kind of popular optimization paradigm for solving hard combinatorial optimization problems. Since randomness is a significant feature of ACO algorithm, it is difficult to analyze and understand ACO from the theoretical point of view. In this paper, we contribute the running time analysis of a simple ACO algorithm, called MMAS*, on the maximum independent set problem (MISP) which is NP-complete. First, we present upper bounds on the running time of MMAS* on the maximum independent set problem with different pheromone values and heuristic information. Then, we show that MMAS* can obtain an approximation ratio of $\frac{\Delta+1}{2}$ on this problem in expected polynomial time, where Δ is the maximum degree in the graph. Finally, the theoretical analysis on an instance shows that MMAS* outperforms the local search algorithm.

Keywords Ant colony optimization, approximation performance, maximum independent set, runtime analysis

1 Introduction

Inspired by natural evolution and biological behavior, researchers have developed many successful bio-inspired algorithms. Ant colony optimization (ACO) is one of the most successful bio-inspired computing methods for complex optimization problems. ACO algorithm derives from the food searching ability of the real ant colonies that can find the shortest path from their nest to a food source. By mimicking the foraging behavior of real ants, Dorigo et al. [1] first proposed the ACO algorithm. The ACO algorithm builds solutions for a given problem by carrying out random walks on a so-called construction graph, and ACO algorithm has been applied to many combinatorial optimization problems and the real industrial problems.

In contrast to the wide range of applications, the theoretical understanding of this kind of algorithms lagged far behind [2]. Therefore, it is desirable and necessary to improve the theoretical foundation of the algorithm in order to have a better understanding of the execution mechanism of the algorithm and guide the algorithm design. Many researches are devoted to understanding the working principles of bio-inspired algorithms, and try to bridge the gap between theoretical research and practical applications of the algorithms. Many encouraging results have been obtained [3].

The initial theoretical studies of ACO algorithms mainly focus on its convergence properties [4]. After that, the focus of theoretical studies turn to the running time analysis. The running time analysis of ACO algorithm is first carried out on some artificial pseudo-boolean functions. Neumann and Witt [5] first investigated the running time of a simple ACO algorithm, called 1-ANT, on the simple pseudo-Boolean function OneMax. Afterwards, some other various pseudo-boolean functions are analyzed [6–9]. For the combinatorial optimization problems, several running time analysis results of ACO have been obtained. Neumann and Witt [10] presented an analysis of 1-Ant for the well-known minimum spanning tree problem. Attiratanasunthron and Fakcharoenphol [11] analyzed the running time of a multiple ant optimization algorithm for the shortest path problem in directed acyclic graphs, and Horoba and Sudholt [12] also presented the running time analysis of ACO algorithm for the shortest

Received; accepted

E-mail: *scutxxy@gmail.com*

path problem, where they extended the results from acyclic graph to arbitrary graph.

Both minimum spanning tree problem and shortest path problem are in P class and can be solved in polynomial time by ACO algorithm. However, there are more interesting combinatorial optimization problems which are NP-hard or NP-complete. Zhou [13] investigated two simple instances of traveling salesman problem (TSP), which is the first work on NP-hard combinatorial optimization problems by ACO algorithm from the perspective of running time analysis. After that, Kötzing et al. [14] conducted a more in-depth study in TSP instances. Huang et al. [15] analyzed the runtime of ACO by molding the ACO algorithm as an absorbing Markov chain, and the upper bounds of the first-hitting time of ACO on TSP are obtained. Peng et al. [16] analyzed the approximation performance of ACO on an NP-complete combinatorial optimization problem, namely TSP(1,2) problem which is a special case of the traveling salesman problem where the weights of all edges are either 1 or 2. Pat [17] carried on the investigation of an ACO on the hypergraph covering problem which is NP-hard. The author investigated the expected runtime of ACO for the minimum weight edge cover problem on generalized hypergraph by taking into account the influence of pheromone values and heuristic information separately. Moreover, Xia et al. [18] presented the performance analysis of a simple ACO algorithm called (1+1) MMAA for the Quadratic assignment problem. The results demonstrate that the (1+1) MMAA can obtain some approximation guarantee on this NP-hard problem. Huang et al. [19] proposed a method of running time analysis of EAs for continuous optimization by introducing an average gain model. The proposed method does not depend on the specific realization of the analyzed EAs and the considered optimization problems. Recently, Qian et al. [20, 21] have made great progress in the theoretical study of evolutionary optimization in noisy environment.

The maximum independent set problem (MISP) is a classic combinatorial optimization problem in computer science, which was proved to be NP-complete [22]. Given an undirected graph, the independent set refers to a set of vertices in which any two vertices are disjoint. The aim of the MISP is to find an independent set such that its cardinality is maximized. In the weighted version, each vertex v has a weight $w(v)$ and the aim is to find an independent set with maximum weight. There are many practical applications such as VLSI design, coding theory, classification theory, geometry, etc [23].

Due to the hardness of exact methods for the NP-complete MISP, approximation algorithms have received much attention from the researchers. The approximation methods can obtain a near optimal solution in polynomial running time. It is a challenge for the researchers to find a good method to solve the MISP. Borowiecki and Göring [24] introduced a potential function on the vertex set and defined a family of Greedy Max-type algorithms for the MISP. They showed that any Greedy Max-type algorithm can obtain some approximation guarantees. Andrade, Resende and Werneck [25] presented two fast local search routines for this problem. The swap operators $(1, 2)$-swap and $(2, 3)$-swap are used in the search process to find near-optimum solutions. For the subcubic planar graphs, Malyshev and Sirotkin [26] proved that the MISP is polynomially solvable in the graphs of this certain class. Furthermore, Alekseev and Sorochan [27] proved the solvability of the MISP in polynomial time for some new cases. Bacsó, Lokshtanov, Marx, et al. [28] studied the MISP on P_t-free graphs without containing any induced path on t vertices. They showed that there exist subexponential time algorithms for this problem.

ACO is usually applied to obtain solutions for NP-hard or NP-complete problems in applications. There are many experimental research results of ACO for solving the maximum independent set problem [25, 29]. However, the theoretical results of ACO algorithms as approximation algorithms are still scarce. In this paper, we will conduct a more in-depth study of ACO for the MISP.

More specifically, we investigate the performance of a simple ACO algorithm called MMAS* with different pheromone values and heuristic information on the MISP. Although Ankit [17] showed that the minimum vertex cover and maximum weak-independent set problems are special cases of the minimum weight edge cover problem in [17], we aim to investigate the approximation performance of MMAS* for the MISP. By imitating the local search algorithm, an approximation ratio of $\frac{\Delta+1}{2}$ can be obtained by the MMAS*, and this approximation ratio is the same as that of the local search algorithm, where Δ is the maximum degree in the graph. Further, we construct an instance of the MISP and show that MMAS* outperforms the local search algorithm on it.

The outline of the paper is as follows. In Section 2, we formally introduce the MISP and MMAS*. In Section 3, we obtain upper bounds of MMAS* for the MISP. In Section 4, we analyze the performance guarantee of MMAS* on the MISP. In Section 5, we construct an instance and show that MMAS* beats local search algorithms on this instance. Finally, we come a conclusion in Section 6.

Algorithm 1 MMAS* for the MISP

Input: The pheromones, each edge with a pheromone value of $1/|A|$, an arbitrary initial solution x.
Output: The best-so-far solutions.

```
Initialization: Set τ_{u,v} = 1/|A| for all (u, v) ∈ A, and randomly generate a solution x;
while termination condition does not hold do
    Generate an offspring x' by using a construction procedure in Algorithm 2;
    if f(x') > f(x) then
        Update the pheromone values, and set x := x'.
    end if
end while
```

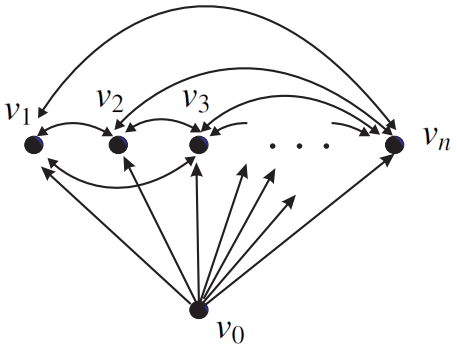

Fig. 1 The construction graph.

2 The MISP and analyzed Algorithms

We start by giving the definition of the MISP, and then introduce the MMAS* algorithm. Given an undirected graph $G(V, E)$ with $|V| = n$ vertices and $|E| = m$ edges, the goal of the MISP is to find a subset $V' \subseteq V$ such that the number of vertices in V' is maximum and any two vertices in V' have no edge in G, i.e., $\forall v_i, v_j \in V', (v_i, v_j) \notin E$.

We investigate a simple ACO algorithm called MMAS* (Min-Max Ant System) (see Algorithm 1) which has been studied in some related theoretical work [9, 13, 14]. MMAS* works iteratively: in the initialization step, each edge gets a pheromone value, then an initial solution x is constructed according to a given directed construction graph $C(G)$. The algorithm constructs a new candidate solution x' in each iteration and compares it with the best-so-far solution. The best-so-far solution is changed if and only if the new solution is strictly better than the current one. At the same time, the pheromone values are updated. The algorithm is described as follows:

We now describe the construction procedure, which has been used in minimum cut problem [30]. In each iteration, a new solution is generated by the random walk of an artificial ant on the construction graph $C(G)$. $C(G)$ is a directed graph, with vertices $\{v_0, v_1, \cdots, v_n\}$, where v_i $(1 \le i \le n)$ corresponds to a vertex of G and v_0 is a starting vertex, as shown in Figure 1. Its edge set A is given by

$$A := \{(v_i, v_j) | 0 \le i \le n, 1 \le j \le n, i \ne j\}.$$

Obviously, $C(G)$ is a complete directed graph by removing the edges pointing to the vertex v_0 and all self-loops.

In the first step, the artificial ant is located at vertex v_0. Then in the following each step, the ant chooses the next node to move according to a probability function, and also the ant avoids visiting two kinds of nodes, i.e., the nodes that are previously visited and the nodes that have been connected to a previously visited node by an edge in E. Without loss of generality, we let v_i be the i-th node traversed. In order to make the random walk of an ant constitutes an independent set, we give a definition of the feasible neighborhood $N(v_i)$ of node v_i, which depends on the previously visited nodes $v_1, v_2, \cdots, v_i$.

$$N(v_i) := (V \backslash \{v_1, v_2, \cdots, v_i\}) \backslash \{u_1 \in V | (u_1, u_2) \in E, u_2 \in \{v_1, v_2, \cdots, v_i\}\}. \quad (1)$$

At each step, the ant selects the next node within its feasible neighborhood according to a probability function, which is given as follows:

$$p(v_{k+1}) = \frac{[\tau_{(v_k, v_{k+1})}]^\alpha \cdot [\eta_{v_{k+1}}]^\beta}{\sum_{u \in N(v_k)} [\tau_{(v_k, u)}]^\alpha \cdot [\eta_u]^\beta}.$$

Thus, the ant located at vertex v_k selects a vertex v_{k+1} with a probability according to the pheromone value τ and heuristic information η on the vertices which are the non-visited successor of v_k. The two positive parameters α and β make a trade-off between the importance of pheromone trails and visibility. The construction procedure on $C(G)$ is described in Algorithm 2.

A solution in the algorithm MMAS* is represented by $x = (x_1, x_2, \ldots, x_n) \in \{0, 1\}^n$, and the bit x_i is set to 1 if vertex v_i is selected, otherwise 0. For the MISP, we need the number of independent points in V to be maximized. Our goal is to maximize the fitness function $f(x)$, which is defined as follows.

$$f(x) = \sum_{i=1}^{n} x_i. \quad (2)$$

Algorithm 2 Construction procedure

Input: A starting search vertex v_0.
Output: An independent set $V^{'}$.
1: Initialization: $V^{'} = \emptyset, E^{'} = \emptyset, v = v_0$;
2: **while** there are nodes that can be added to $V^{'}$ **do**
3: Select one neighbor $v_{k+1} \in N(v_k)$ with probability $\frac{[\tau_{(v_k,v_{k+1})}]^{\alpha}\cdot[\eta_{v_{k+1}}]^{\beta}}{\sum_{u\in N(v_k)}[\tau_{(v_k,u)}]^{\alpha}\cdot[\eta_u]^{\beta}}$;
4: Add node v_{k+1} to $V^{'}$, set $k := k + 1$;
5: **end while**
6: **return** V'.

In the initialization step of Algorithm 1, the pheromone value of each edge $(v, v^{'})$ is set to $1/|A|$ such that the sum of the pheromone values is equal to 1. Thus, for all $(v, v^{'}) \in A$, $\tau_{(v,v^{'})} = 1/|A|$. We denote $\eta_{v^{'}}$ the heuristic information of a vertex $v^{'}$, which is defined as the inverse of the weight of G corresponding to the node v in $C(G)$.

Once the best-so-far solution $x^{'}$ is obtained, the pheromone values are updated according to the following pheromone update formula:

$$\forall(v, v^{'}) \in A : \tau^{'}_{(v,v^{'})} = \begin{cases} h, \text{ if } v^{'} \in x^{'}, \\ l, \text{ otherwise,} \end{cases}$$

where l and h are the lower bound and upper bound of pheromone values on each edge in $C(G)$ and we choose $h = n^2 l$ throughout the paper. Therefore, after a pheromone update procedure, the pheromone values on the edges connected to the nodes in $C(G)$ in the best-so-far solution are h and the remaining edges have pheromone values l. Thus, MMAS* tends to give more larger pheromone to the best-so-far solutions, which makes the algorithm with a greater probability to choose a better solution.

3 Upper bounds of MMAS* on the MISP

In this section, we study the upper bounds on the expected optimization time of MMAS* on the MISP. Both theorems given below are expressed in terms of the pheromone values, the heuristic information, the lower bound, the upper bound and the cardinality of an optimal solution. In Theorem 3.1 we only consider the influence of pheromone values while in Theorem 3.2 we only consider the influence of heuristic information.

Theorem 3.1. *Let control parameters $\alpha = 1$ and $\beta = 0$. For any MISP instance, the expected optimization time of the* MMAS* *is* $O\left(\frac{((n-k)\frac{h}{l}+k)!}{k!}\right)$.

Proof. First, we analyze the lower bound of the probability that the MMAS* obtains an optimal solution. Without loss of generality, we assume that there are k vertices in the optimal solution V^*, and we also pessimistically assume that these edges pointing to vertices of V^* have pheromone level l and all the other edges have pheromone level h.

Let E_i be the event under the premise that the first $i-1$ vertices in V^* have been chosen, in the i-th step an edge pointing to a node in V^* with pheromone level l is chosen. The probability that event E_i occurs is at least

$$\frac{(k-(i-1))l}{(n-k)h + (k-(i-1))l} = \frac{k-(i-1)}{(n-k)\frac{h}{l} + (k-(i-1))}.$$

Then the probability of MMAS* finding an optimal solution is

$$\prod_{i=1}^{k} P(E_i) = \prod_{i=1}^{k} \frac{k-(i-1)}{(n-k)\frac{h}{l} + (k-(i-1))} \geq \frac{k!}{\left((n-k)\frac{h}{l}+k\right)!}.$$

Therefore, we can obtain the expected optimization time

$$O\left(\frac{\left((n-k)\frac{h}{l}+k\right)!}{k!}\right).$$

□

In the following, we analyze the expected optimization time of the MMAS* by only considering the influence of heuristic information.

Theorem 3.2. *Let control parameters $\alpha = 0$ and $\beta = 1$. For any MISP instance, the expected optimization time of the* MMAS* *is* $O\left(\frac{n!}{k!}\right)$.

Proof. Let $M = \{\eta_1, \eta_2, \cdots, \eta_n\}$ be a set of the heuristic information of all vertices in $C(G)$ and $\eta_1, \eta_2, \cdots, \eta_n$ denote the heuristic information values of $v_1, v_2, \cdots, v_n$, respectively. Without loss of generality, we suppose that the optimal solution is $V^* = \{v_1, v_2, \cdots, v_k\}$ and the corresponding heuristic information is $N = \{\eta_1, \eta_2, \cdots, \eta_k\}$. We also denote N_i the set containing i elements of N.

Let E_i be the event under the premise that the first $i-1$ vertices in V^* have been chosen, in the i-th step an edge pointing

to a node in V^* with heuristic information value of h is chosen. Then

$$\begin{aligned}
&P(E_i)\\
&= \frac{\sum\limits_{\eta_j \in N \setminus N_{i-1}} \eta_j}{\sum\limits_{\eta_i \in M \setminus N_{i-1}} \eta_j}\\
&= \frac{\sum\limits_{\eta_j \in N \setminus N_{i-1}} \eta_j}{\sum\limits_{\eta_j \in M \setminus N} \eta_j + \sum\limits_{\eta_j \in N \setminus N_{i-1}} \eta_j}\\
&= \frac{\eta_{j_1} + \eta_{j_2} + \cdots + \eta_{j_{k-(i-1)}}}{(\eta_{k+1} + \eta_{k+1} + \cdots + \eta_n) + (\eta_{j_1} + \eta_{j_2} + \cdots + \eta_{j_{k-(i-1)}})}\\
&\geq \frac{(k-(i-1))\eta_{min}}{(k-(i-1))\eta_{min} + (n-k)\eta_{max}}\\
&= \frac{k-(i-1)}{k-(i-1) + (n-k)\frac{\eta_{max}}{\eta_{min}}},
\end{aligned}$$

where η_{max} and η_{max} are the maximum heuristic information and the minimum heuristic information of edges in $C(G)$, respectively. Hence, we can obtain the lower bound of the probability that MMAS* obtains an optimal solution.

$$\begin{aligned}
\prod_{i=1}^{k} P(E_i) &= \prod_{i=1}^{k} \frac{k-(i-1)}{k-(i-1) + (n-k)\frac{\eta_{max}}{\eta_{min}}}\\
&\geq \frac{k!}{\left((n-k)\frac{\eta_{max}}{\eta_{min}} + k\right)!}.
\end{aligned}$$

Therefore, the expected optimization time of MMAS* on any MISP instance is

$$O\left(\frac{\left((n-k)\frac{\eta_{max}}{\eta_{min}} + k\right)!}{k!}\right).$$

The above results are adapted to the maximum independent set problem with weight. It is easy to see that for MISP, we let $\eta_{max} = \eta_{min}$ and this gives the desired expected optimization time. □

The above two theorems imply that MMAS* is hard to obtain an optimal solution in expected polynomial time. This is mainly because that the MISP is NP-complete and it is hard to be solved in polynomial time. Therefore, we can only expect to get its approximate optimal solution.

4 Approximation performance of MMAS* on the MISP

Now we turn to the approximation performance of MMAS* on the MISP. First, we introduce a fitness partitioning method, called fitness-level method estimation, to estimate the running time of MMAS*. The fitness-level method is an efficient mathematical tool for the theoretical analysis of evolutionary algorithm [31]. In order to give the definition of fitness-level method, we first give the definition of f-partition.

Definition 1. (f-partition) Let fitness function $f : S \to R$ be a function that should be maximized, where S is a finite search space. f can obtain $m+1$ different values $f_0, f_1, \ldots, f_m$ which satisfy $f_0 < f_1 < \ldots, < f_m$. We define $A_i = \{x \in S | f(x) = f_i\}$ $(i = 0, 1, \ldots, m)$. We call the set $\{A_0, A_1, \ldots, A_m\}$ an f-partition.

Based on the definition of f-partition, we can obtain an upper bound technique called fitness-level method, which provides the expected running time of the simple ACO algorithm.

Lemma 4.1. *Let set $\{A_0, A_1, \ldots, A_m\}$ be an f-partition. Let $p(x)$ denote the probability that $x \in A_i$ is mutated into $A_{i+1} \cup A_{i+2} \cup \ldots \cup A_m$, and let $p_i = min\{p(x)|x \in A_i\}$. Then the expected optimization running time for ACO algorithm to find a global optimal search point is upper bounded by*

$$p_1^{-1} + p_2^{-1} + \ldots + p_{m-1}^{-1}.$$

Proof. See [2]. □

The MISP is an NP-complete problem, which cannot be solved in polynomial time. Therefore, we usually expect to obtain an approximate optimal solution. For the MISP in bounded degree graphs, Khanna et al. [32] obtained an approximation ratio by using a local search algorithm around 3-flip neighborhood.

In the following, we will prove that by mimicking the local search algorithm around 3-flip neighborhood, the MMAS* can obtain an approximation ratio the same as that obtained by the local search algorithm with 3-flip neighborhood.

Lemma 4.2 ([32])**.** *For any instance of the MISP, in which the degree of any vertex is bounded by a constant Δ, the local search algorithm around 3-flip neighborhood obtains an approximation ratio of $\frac{\Delta+1}{2}$.*

Now we introduce the definitions of 3-flip operation and the local search algorithm around 3-flip neighborhood. Let V be an independent set. A 3-flip operation transforms V to another independent set $V^{'}$ by adding a vertex to V or removing a vertex in V, and at the same time adding two vertices to V which are not in V. The local search algorithm is an iterative

improvement method, and at each step, the current solution is improved according to its neighbors. Let $S^{'}$ be a new independent set obtained by performing a 3-flip operation on S, then the local search algorithm with 3-flip neighborhood works as follows. In the first step, the algorithm gives an initial solution S_0, then in each subsequent iteration, algorithm transforms solution S to $S^{'}$ by using the 3-flip operation in its neighbours. This process stops if the solution cannot be improved.

In order to obtain an approximation ratio of MMAS* on the MISP, we first investigate the expected optimization time of MMAS* performing a specific 3-flip operation with respect to certain parameter settings.

Lemma 4.3. *For any MISP instance with optimal solution including $\Theta(n)$ points, let control parameters $\alpha = 1$ and $\beta = 0$, then the expected optimization time of the* MMAS* *performing a specific 3-flip operation is $O(\frac{n^4}{\ln^2 n})$.*

Proof. Suppose that the MMAS* has obtained an independent set V^{last} with $|V^{last}| = k^{'}$ in the last accepted solution, where $k^{'} = cn$ and c is a constant with $0 \leq c \leq 1$. Let $V = V^{last} \setminus \{v\} \cup \{v^{'}, v^{''}\}$ be any independent set that is obtained from V^{last} by removing one vertex v in V^{last} and adding another two vertices $v^{'}, v^{''}$ which are not in V^{last}. We let E be the event of transforming V^{last} to V, and in the following we will prove that the probability of event E occurring is $\Omega(\frac{\ln^2 n}{n^4})$.

Without loss of generality, we let $V^{last} = \{v_1, \ldots, v_{k^{'}}\}$ and suppose that $V = \{v_1, \ldots, v_{k^{'}-1}, v^{'}, v^{''}\}$, where $v = v_{k^{'}}$,$v^{'} \neq v_i$ and $v^{''} \neq v_i$ for $1 \leq i \leq k^{'}$. We denote $E_{i,j}$ the event that the i-th node and the j-th node visited by MMAS* are $v^{'}$ and $v^{''}$, respectively, whereas the rest $k^{'} - 1$ nodes are from $\{v_1, \ldots, v_{k^{'}-1}\}$. We divide the event $E_{i,j}$ into two parts $E^1_{i,j}$ and $E^2_{i,j}$. $E^1_{i,j}$ occurs if and only if the i-th node and the j-th node visited by MMAS* are arbitrary whereas the rest $k^{'} - 1$ nodes are from $\{v_1, \ldots, v_{k^{'}-1}\}$, and $E^2_{i,j}$ occurs if and only if the i-th node and the j-th node visited by MMAS* are $v^{'}$ and $v^{''}$, respectively, whereas the rest $k^{'} - 1$ nodes are arbitrary. Obviously, $E_{i,j} = E^1_{i,j} \bigcap E^2_{i,j}$.

Note that the edges in $C(G)$ which are pointing to nodes of V^{last} have high pheromone value h while the rest of edges have low value l. We consider the condition that $k-1$ nodes of V^{last} have been included, then the probability of not selecting a vertex of V^{last} which is connected to a visited node in $C(G)$ is

$$\begin{aligned}
&\frac{(n-k^{'})l}{(n-k^{'})l + (k^{'}-k+1)h} \\
&\leq \frac{(n-k^{'})l}{(k^{'}-k+1)h} \\
&= \frac{(n-cn)l}{(cn-k+1)h} \quad \text{(by using } k^{'} = cn\text{)} \\
&\leq \frac{1}{(cn-k+1)n}.
\end{aligned}$$

In the last equality, we have used the equation $h = n^2 l$. Then we have

$$\begin{aligned}
P(E^1_{i,j}) &= \prod_{k=1}^{i-1}\left(1 - \frac{1}{(cn-k+1)n}\right) \cdot \prod_{k=i+1}^{j-1}\left(1 - \frac{1}{(cn-k+2)n}\right) \\
&\quad \cdot \prod_{k=j+1}^{cn}\left(1 - \frac{1}{(cn-k+3)n}\right) \\
&\geq \left(1 - \frac{1}{n}\right)^{cn-2} \\
&= \Omega(1),
\end{aligned}$$

where we assume $i < j$ with $j - i > 1$, and if $j - i = 1$ we can also obtain the same conclusion.

For the event $E^2_{i,j}$, the probability of selecting the i-th node which is $v^{'}$ is equal to

$$\begin{aligned}
&\frac{l}{(n-k^{'})l + (k^{'}-i+1)h} \\
&= \frac{l}{(n-k^{'})l + (k^{'}-i+1)n^2 l} \\
&= \Omega\left(\frac{1}{(cn-i+1)n^2}\right).
\end{aligned}$$

In a similar way, the probability of selecting the j-th node which is $v^{''}$ is equal to

$$\begin{aligned}
&\frac{l}{(n-k^{'}-1)l + (k^{'}-j+2)h} \\
&= \frac{l}{(n-k^{'}-1)l + (k^{'}-j+2)n^2 l} \\
&= \Omega\left(\frac{1}{(cn-j+2)n^2}\right).
\end{aligned}$$

Since $E_{i,j} = E^1_{i,j} \bigcap E^2_{i,j}$, we can obtain that $P(E_{i,j}) = P(E^1_{i,j})P(E^2_{i,j})$.

Note that the events $E_{i,j}$ are mutually disjoint events, then the probability of transforming V^{last} to V is

$$
\begin{aligned}
P(E) &= P(\sum_{j=1,j\neq i}^{k'+1}\sum_{i=1}^{k'} E_{i,j}) \\
&\geq \sum_{j=1,j\neq i}^{k'+1}\sum_{i=1}^{k'} P(E_{i,j}) \\
&= \sum_{j=1,j\neq i}^{k'+1}\sum_{i=1}^{k'} P(E^1_{i,j})\cdot P(E^2_{i,j}) \\
&= \sum_{j=1,j\neq i}^{k'+1} \Omega\left(\frac{1}{(cn-j+2)n^2}\right)\cdot \sum_{i=1}^{k'} \Omega\left(\frac{1}{(cn-i+1)n^2}\right) \\
&= \Omega\left(\sum_{j=1,j\neq i}^{k'+1} \frac{1}{(cn-j+2)n^2}\right)\cdot \Omega\left(\sum_{i=1}^{k'} \frac{1}{(cn-i+1)n^2}\right) \\
&= \Omega(\frac{\ln n}{n^2})\cdot\Omega(\frac{\ln n}{n^2}) \\
&= \Omega(\frac{\ln^2 n}{n^4}).
\end{aligned}
$$

□

Theorem 4.4. *For any given instance of the MISP with a bound of constant Δ on the degree of any vertex, let opt be the cardinality of vertices in the optimal solution and let $opt = \Theta(n)$. The MMAS* can find an independent set with approximation ratio of $\frac{\Delta+1}{2}$ in expected running time $O(\frac{n^5}{\ln^2 n})$.*

Proof. We partition the search space $\{0,1\}^n$ into two disjoint sets A_1, A_2 with respect to different fitness values as follows.

$$
\begin{aligned}
A_1 = \{x \in \{0,1\}^n | x \text{ is a feasible solution} \\
\text{and satisfies } 0 \leq f(x) < \frac{2opt}{\Delta+1}\}, \\
A_2 = \{x \in \{0,1\}^n | x \text{ is a feasible solution} \\
\text{and satisfies } f(x) \geq \frac{2opt}{\Delta+1}\},
\end{aligned}
$$

where *opt* is also the fitness value of an optimal solution. Note that during the operation of the MMAS*, the fitness value $f(x)$ of the current solution x never decreases.

Without loss of generality, we suppose that the current solution x belongs to A_1. According to Lemma 4.2, since $f(x) < \frac{2opt}{\Delta+1}$, then there must exist at least one feasible solution in its 3-flip neighborhood that the fitness value can be increased by at least one. Following the Lemma 4.3, the probability of the MMAS* performing a specific 3-flip operation is $\Omega(\frac{\ln^2 n}{n^4})$. As long as the current solutions belong to A_1, the algorithm needs to perform the 3-flip operation sequentially. Note that when x belongs to A_1, then we have $0 \leq f(x) \leq \frac{2opt}{\Delta+1} \leq n$. Therefore, after at most n 3-flip operations, the MMAS* will find a solution in A_2. Thus, the expected running time for the MMAS* starting from A_1 to find a solution with fitness value at least $\frac{2opt}{\Delta+1}$ is bounded above by $O(\frac{n^5}{\ln^2 n})$, which completes the proof. □

5 The MMAS* beats the local search algorithms on a MISP instance

A local search algorithm, as mentioned before, is started from an arbitrary initial solution, and then finds an improved solution in the neighborhood of the current solution until no any such improved solution can be found. Due to the limitation of the search neighborhood space, the local search algorithms are likely to get trapped in a local optimum. The ACO algorithm is generally considered to be a global optimization method that can jump out of local optimum.

In this section, we illustrate this idea by constructing an instance $G_1 = (V, E)$ of the MISP shown in Figure 2, where V consists of two sets V_1 and V_2. $V_1 = \{u_1, u_2, \cdots, u_k\}$ and $V_2 = \{v_1, v_2, \cdots, v_k, v_{1,2}, v_{1,3}, \cdots, v_{1,k}, v_{2,3}, v_{2,4}, \cdots, v_{2,k}, \cdots, v_{k-1,k}\}$. Hence, the set V_1 has k elements and the set V_2 has $k + \frac{k(k-1)}{2}$ elements. The total number of vertices is $n = k + k + \frac{k(k-1)}{2}$, thus, $k = \Theta(n^{\frac{1}{2}})$. For all $1 \leq i \leq k$, any vertex u_i in V_1 is connected to any vertex in V_2 which contains the subscript i by an edge, all of which constitute the edge set E of G_1.

For G_1, a solution that contains all the elements in V_2 but no elements in V_1 is a global optimum, and a solution that contains all elements in V_1 but no elements in V_2 is called a local optimum with respect to the local search algorithm around 3-flip neighborhood. This is because that the execution of any single 3-flip operation can not improve the current solution, and makes the local search algorithm around 3-flip neighborhood falling into local optimum. However, the ACO algorithms have the potential of avoiding being trapped into local optimum.

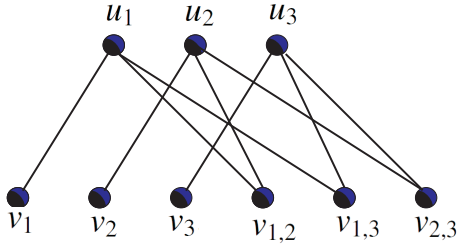

Fig. 2 Instance G_1 with $k = 3$.

Lemma 5.1. *Suppose that control parameters $\alpha = 1$ and $\beta = 0$, then the expected running time for the MMAS* to jump out of the local optimum of instance G_1 is $O\left(\frac{n^5}{\ln^3 n}\right)$.*

Proof. The idea behind the proof is similar to that of Lemma 4.3. For instance G_1, if the current solution consists of all elements in V_1, then it is a local optimum. The MMAS* only accepts the event of adding elements v_i, v_j and $v_{i,j}(i < j)$ from V_2 and simultaneously deleting u_i and u_j from V_1. We denote this event by E, and show that the probability of this event is $\Omega\left(\frac{\ln^3 n}{n^5}\right)$.

Recall that $V_1 = \{u_1, u_2, \cdots, u_k\}$ and let $V = V_1 \setminus \{u_i, u_j\} \cup \{v_i, v_j, v_{i,j}\}$. We let $E_{i,j,h}$ be the event that the i-th node, the j-th node and the h-th node visited by MMAS* are v_i, v_j and $v_{i,j}$, respectively, whereas the rest $k-2$ nodes are from $V_1 \setminus \{u_i, u_j\}$.

We divide the event $E_{i,j,h}$ into two parts $E^1_{i,j,h}$ and $E^2_{i,j,h}$ $(i < j)$. $E^1_{i,j,h}$ occurs if and only if the i-th node, the j-th node and the h-th node visited by MMAS* are arbitrary whereas the rest $k-2$ nodes are from $V_1 \setminus \{u_i, u_j\}$. $E^2_{i,j,h}$ occurs if and only if the i-th node, the j-th node and the h-th node visited by MMAS* are v_i, v_j and $v_{i,j}$, respectively, whereas the rest $k-2$ nodes are arbitrary.

We consider the condition that $s-1$ nodes of $V_1 \setminus \{u_i, u_j\}$ have been included, then the probability of not selecting a node of V_1 as the next node visited is

$$\begin{aligned}\frac{(n-k)l}{(n-k)l+(k-s+1)h} &\le \frac{(n-k)l}{(k-s+1)h}\\ &= \frac{(n-k)l}{(k-s+1)n^2 l}\\ &\le \frac{1}{(k-s+1)n}.\end{aligned}$$

In the last equality, we have used the equation $h = n^2 l$. Then we have

$$\begin{aligned}P(E^1_{i,j,h}) &= \prod_{s=1}^{i-1}\left(1-\frac{1}{(k-s+1)n}\right)\cdot\prod_{s=i+1}^{j-1}\left(1-\frac{1}{(k-s+2)n}\right)\\ &\quad\cdot\prod_{s=j+1}^{h-1}\left(1-\frac{1}{(k-s+3)n}\right)\cdot\prod_{s=h+1}^{k}\left(1-\frac{1}{(k-s+4)n}\right)\\ &\ge\left(1-\frac{1}{n}\right)^{k-3}\\ &= \Omega(1),\end{aligned}$$

where we assume $i < j < h$ with $j-i > 1$ and $h-j > 1$, and if $j-i = 1$ or $h-j = 1$ we can also obtain the same conclusion.

Now we turn to the event $E^2_{i,j,h}$. We denote the event of selecting the i-th node which is v_i by $E^{2,i}_{i,j,h}$, then

$$\begin{aligned}P(E^{2,i}_{i,j,h}) &= \frac{l}{(n-k)l+(k-i+1)h}\\ &= \frac{l}{(n-k)l+(k-i+1)n^2 l}\\ &= \Omega\left(\frac{1}{(k-i+1)n^2}\right).\end{aligned}$$

In a similar way, we denote the event of selecting the j-th node and the h-node that are v_j and $v_{i,j}$ by $E^{2,j}_{i,j,h}$ and $E^{2,h}_{i,j,h}$, respectively. Obviously, $E^2_{i,j,h} = E^{2,i}_{i,j,h} \bigcap E^{2,j}_{i,j,h} \bigcap E^{2,h}_{i,j,h}$.

Note that the events $E_{i,j,h}$ are mutually disjoint events and points v_i and v_j are arbitrary, then the probability of transforming V_1 to V is

$$\begin{aligned}P(E) &= P\left(\binom{k}{2}\cdot\sum_{h=1,h\neq i,j}^{k}\sum_{j=1,j\neq i}^{k}\sum_{i=1}^{k}E_{i,j,h}\right)\\ &= \binom{k}{2}\cdot\sum_{h=1,h\neq i,j}^{k}\sum_{j=1,j\neq i}^{k}\sum_{i=1}^{k}P\left(E_{i,j,h}\right)\\ &= \frac{k(k-1)}{2}\cdot\sum_{h=1,h\neq i,j}^{k}\sum_{j=1,j\neq i}^{k}\sum_{i=1}^{k}\Omega\left(\frac{1}{(k-i+1)n^2}\right)\\ &\quad\cdot\Omega\left(\frac{1}{(k-j+1)n^2}\right)\cdot\Omega\left(\frac{1}{(k-h+1)n^2}\right)\\ &= \frac{k(k-1)}{2}\cdot\Omega\left(\sum_{h=1,h\neq i,j}^{k}\frac{1}{(k-h+1)n^2}\right)\\ &\quad\cdot\Omega\left(\sum_{j=1,j\neq i}^{k}\frac{1}{(k-j+1)n^2}\right)\cdot\Omega\left(\sum_{i=1}^{k}\frac{1}{(k-i+1)n^2}\right)\\ &= \Omega\left(k^2\cdot\left(\frac{\ln k}{n^2}\right)^3\right)\\ &= \Omega\left(\frac{\ln^3 n}{n^5}\right).\end{aligned}$$

This completes the proof. □

Theorem 5.2. *Suppose that the control parameters $\alpha = 1$ and $\beta = 0$. Starting from an any initial solution, the MMAS* can efficiently find the global optimum in expected running time $O\left(\frac{n^6}{\ln^3 n}\right)$.*

Proof. We divide the proof into two parts. In the first part, when the current solution contains non-independent vertices. The algorithm will accept the operation of removing a non-independent vertex. In the second part, when the current solution only contains independent vertices and we can divide this part into four cases:

Case 1: x is a local optimal solution. In this case, according to Lemma 4, the event of deleting two elements $\{u_i\}$ and

$\{u_j\}$ from V_1, and simultaneously adding three elements $\{v_i\}$, $\{v_j\}$ and $\{v_{i,j}\}$ in V_2 to jump out of the local optimal solution.

Case 2: x only contains vertices in V_1, and the number of vertices is less than k. In this case, the fitness value can be improved by adding one vertex in V_1 or adding one vertex that is in V_2 but not connected to V_1.

Case 3: x contains both vertices in V_1 and V_2, and the number of vertices in V_1 is less than k. In this case, changing three vertices can improve the fitness value.

Case 4: x only contains the vertices in V_2. In this case, adding any vertex in V_2 can improve the fitness value.

According to the above cases, we can see that the probability of the event in case 1 is the smallest. The probability of the events in other cases is at least $\frac{1}{n}$ larger than that of Case 1. Note that according to Lemma 5.1, the probability of the event in case 1 is $\Omega\left(\frac{\ln^3 n}{n^5}\right)$. Therefore, the MMAS* can find the global optimum in expected running time $O\left(\frac{n^6}{\ln^3 n}\right)$. □

6 Conclusions

In the present work, we contribute to the theoretical understanding of a kind of ACO algorithm by investigating the classic maximum independent set problem. Our theoretical results show that with a new construction graph, the ACO algorithm can obtain an approximation ability on maximum independent set problem, and also show the impact of the parameter settings. We first obtain two general upper bounds on arbitrary maximum independent set instance, then we obtain an approximation ratio by ACO algorithm in polynomial time. Finally, we give an instance on which ACO algorithm can escape from local optimum in polynomial time while the local search algorithm is easy to get stuck in local optimum.

In future work, there are several directions to be done. Considering other different construction graph for the maximum independent set problem; Making in-depth analysis of the impact of pheromone value on the running time; Extending the running time analysis to pheromone evaporation factor; Extending the present work to other NP-hard combinatorial optimization problems. How to use the drift analysis method [33] and switch analysis [34] for the investigation on the runtime of ACO is another interesting work.

Acknowledgements This work is supported by the National Natural Science Foundation of China (Grant Nos. 61703183, 61773410 and 61876207), the Public Welfare Technology Application Research Plan of Zhejiang Province (LGG19F030010), and the Science and Technology Program of Guangzhou (202002030260).

References

1. Dorigo M, Stützle T. Ant Colony Optimization. Cambrigde MA: MIT Press, 2004
2. Neumann F, Witt C. Bioinspired Computation in Combinatorial Optimization - Algorithms and Their Computational Complexity. Berlin Heidelberg: Springer, 2010
3. Zhou Z H, Yu Y, Qian C. Evolutionary Learning: Advances in Theories and Algorithms. Singapore: Springer, 2019
4. Gutjahr W J. ACO algorithms with guaranteed convergence to the optimal solution. Information Processing Letters, 2002, 82(3): 145-153
5. Neumann F, Witt C. Runtime analysis of a simple ant colony optimization algorithm. Algorithmica, 2009, 54(2): 243-255
6. Gutjahr W J. First steps to the runtime complexity analysis of ant colony optimization. Computers & Operations Ressearch, 2008, 35(9): 2711-2727
7. Doerr B, Johannsen D. Refined runtime analysis of a basic ant colony optimization algorithm. In: Proceedings of IEEE Congress on Evolutionary Computation, 2007: 501-507
8. Doerr B, Neumann F, Sudholt D, Witt C. On the runtime analysis of the 1-ANT ACO algorithm. In: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation Conference. 2007: 33-40
9. Neumann F, Sudholt D, Witt C. Analysis of different MMAS ACO algorithms on unimodal functions and plateaus. Swarm Intelligence, 2009, 3(1): 35-68
10. Neumann F, Witt C. Ant colony optimization and the minimum spanning tree problem. Theoretical Computer Science, 2010, 411(25): 2406-2413
11. Attiratanasunthron N, Fakcharoenphol J. A running time analysis of an ant colony optimization algorithm for shortest paths in directed acyclic graphs. Information Processing Letters, 2008, 105(3): 88-92
12. Sudholt D, Thyssen C. Running time analysis of Ant Colony Optimization for shortest path problems. Journal of Discrete Algorithms, 2012, 10: 165-180
13. Zhou Y. Runtime analysis of an ant colony optimization algorithm for TSP instances. IEEE Transactions on Evolutionary Computation, 13(5), 2009, 1083-1092
14. Kötzing T, Neumann F, Roglin H. Theoretical analysis of two ACO approaches for the traveling salesman problem. Swarm Intelligence, 2012, 6: 1-21
15. Huang H, Wu H, Zhang Y, Lin Z, Hao Z F. Running time Analysis of Ant System Algorithms with Upper-bound Comparison. International Journal of Swarm Intelligence Research, 2017, 8(4), 1-17
16. Peng X, Zhou Y, Xu G. Approximation performance of Ant Colony Optimization for the TSP(1,2) problem. International Journal of Computer Mathematics, 2016, 93(10): 1683-1694
17. Pat A. Ant Colony Optimization and Hypergraph Covering Problems. In: Proceedings of the IEEE Congress on Evolutionary Computation. 2014, 1714-1720
18. Xia X, Zhou Y. Performance Analysis of ACO on the Quadratic As-

signment Problem. Chinese Journal of Electronics, 2018, 27(1): 26-34
19. Huang H, Su J, Zhang Y, Hao Z. An Experimental Method to Estimate Running Time of Evolutionary Algorithms for Continuous Optimization. IEEE Transactions on Evolutionary Computation, 2020, 24(2): 275-289
20. Qian C. Distributed Pareto Optimization for Large-scale Noisy Subset Selection. IEEE Transactions on Evolutionary Computation, DOI: 10.1109/TEVC.2019.2929555
21. Qian C, Yu Y, Zhou Z H. Analyzing Evolutionary Optimization in Noisy Environments. Evolutionary Computation, 2018, 26(1): 1-41
22. Garey M R, Johnson D S. Computers and Intractability: A guide to the theory of NP-completeness. San Francisco: W. H. Freeman, S1979
23. Das K N, Chaudhuri B. Heuristics to Find Maximum Independent Set: An Overview. In: Proceedings of the International Conference on Soft Computing for Problem Solving, Advances in Intelligent and Soft Computing. 2011, 881-892
24. Borowiecki P, Göring F. GreedyMAX-type Algorithms for the Maximum Independent Set Problem. In: Proceedings of the 37th Conference on Current Trends in Theory and Practice of Computer Scienc. 2011, 146-156
25. Andrade D V, Resende M G C, Werneck R F. Fast local search for the maximum independent set problem. Journal of Heuristics, 2012, 18(4): 525-547
26. Malyshev D S, Sirotkin D V. Polynomial-time solvability of the independent set problem in a certain class of subcubic planar graphs. Journal of Applied and Industrial Mathematics, 2017, 11(3): 400-414
27. Alekseev V E, Sorochan S V. New Cases of the Polynomial Solvability of the Independent Set Problem for Graphs with Forbidden Paths. Journal of Applied and Industrial Mathematics, 2018, 12(2): 213-219
28. Bacsó G, Lokshtanov D, Marx D, et al. Subexponential-Time Algorithms for Maximum Independent Set in P_t-Free and Broom-Free Graphs. Algorithmica, 2019, 81(2): 421-438
29. Li Y, Xu Z. An ant colony optimization heuristic for solving maximum independent set problems. In: Proceedings Fifth International Conference on Computational Intelligence and Multimedia Applications. 2003, 206-211
30. Kötzing T, Lehre P K, Neumann F, Oliveto P S. Ant colony optimization and the minimum cut problem. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation Conference. 2010: 1393-1400
31. Droste S, Jansen T, Wegener I. On the analysis of the (1+1)evolutionary algorithm. Theoretical Computer Science, 2002, 276: 51-81
32. Khanna S, Motwani R, Sudan M, Vazirani U. On syntactic versus computational views of approximability. In: Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science. 1994, 819-836
33. He J, Yao X. Average drift analysis and population scalability. IEEE Transactions on Evolutionary Computation, 2017, 21(3): 426-439
34. Yu Y, Qian C, Zhou Z H. Switch analysis for running time analysis of evolutionary algorithms. IEEE Transactions on Evolutionary Computation, 2015, 19(6): 777-792