THE UNIVERSITY OF AUCKLAND
LIBRARIES AND LEARNING SERVICES

ResearchSpace@Auckland

Version
This is the Accepted Manuscript version. This version is defined in the NISO recommended practice RP-8-2008 http://www.niso.org/publications/rp/

Suggested Reference

Dinneen, M. J., & Wei, K. (2013). A (1+1) adaptive memetic algorithm for the maximum clique problem. In *IEEE Congress on Evolutionary Computation (CEC) 2013* (pp. 1626-1634). Cancun, Mexico. doi:10.1109/CEC.2013.6557756

Copyright

Items in ResearchSpace are protected by copyright, with all rights reserved, unless otherwise indicated. Previously published items are made available in accordance with the copyright policy of the publisher.

© 2013 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

http://www.ieee.org/publications_standards/publications/rights/rights_policies.html

https://researchspace.auckland.ac.nz/docs/uoa-docs/rights.htm

A (1+1) Adaptive Memetic Algorithm for the Maximum Clique Problem

Michael J. Dinneen and Kuai Wei*
Department of Computer Science, University of Auckland, Auckland, New Zealand
{mjd,kuai}@cs.auckland.ac.nz

*Corresponding author.

***Abstract*—A memetic algorithm (MA) is an Evolutionary Algorithm (EA) augmented with a local search. We previously defined a *(1+1) Adaptive Memetic Algorithm* (AMA) with two different local searches, and the comparison with the well-known (1+1) EA, Dynamic (1+1) EA and (1+1) MA on some toy functions showed promise for our proposed algorithm.**

In this paper we focus on the NP-hard Maximum Clique Problem, and show the success of our proposed (1+1) AMA. We propose a new metric (expected running time to escape a local optimal), and show how this metric dominates the expected running time of finding a maximum clique. Then based on this new metric, we show the above analyzed algorithms are expected to find a maximum clique on graphs, bipartite graphs and sparse random graphs in a polynomial time in the number of vertices.

Also based on our new metric, we will show that if an algorithm takes an exponential time to find a maximum clique of a graph, it must have been trapped into at least one local optimal that is extremely hard to escape. Furthermore, we will show that our proposed (1+1) AMA with a random permutation local search is expected to escape these (hard to escape) local optimal cliques drastically faster than the well-known basic (1+1) EA. The success of our experimental results also shows the benefit of our adaptive strategy combined with the random permutation local search.

***Keywords*—memetic algorithms, maximum clique problem**

I. Introduction

A Memetic Algorithm (MA) is a meta-heuristic algorithm which hybridizes Evolutionary Algorithms (EAs) and local searches. This hybridization preserves both the exploratory search ability of evolutionary algorithms and the neighborhood search ability of local searches [1], which has led to many interests in MAs. The successful experimental results have showed the merits of MAs. An overview in 2011 shows the usefulness of MAs in many applications [13].

However, theoretical studies of MAs are far behind experimental studies. Since MAs combine EAs and local searches, to study the theory of MAs, we first study EAs. There are a number of theoretical investigations on EAs in the literature. A survey can be found in [16]. In short, the time complexity analysis of EAs started from the basic (1+1) EA (an EA with only a mutation approach but no crossover) on simple pseudo-boolean functions [20] in 1998, Onemax [20] in 1998, Trap Functions [3] in 1998, and plateaus of constant fitness [8] in 2001. In 2002, Droste, Jansen and Wegener [4] summarized the basic (1+1) EA, where most theoretic studies of EAs are based on this algorithm. A very important progress is the analysis on population-based EAs, such as the $(\mu + 1)$ EA [25], $(1 + \lambda)$ EA [7]. Also many researches have focused on showing the crossover operation is essential in EAs such as [9], [12], [19].

Meantime, after the basic (1+1) EA were analyzed, some variants of EAs, including MAs, have been formalized and analyzed on some artificially created functions. These studies help us understand what characteristics of these algorithms may make their optimizations easier or harder than the basic (1+1) EA. Examples are the Dynamic (1+1) EA [10] in 2006, (1+1) MA [22] in 2006, (1+1) Genetic Programming [5] in 2011 and our (1+1) AMA [2] in 2013. Apart from analyzing those toy functions (`ONEMAX`, `BIN`, and `LEADINGONES`, etc.), many researches have started to analyze the EAs for main-stream combinatorial optimization problems such as the Maximum Matching Problem [6] in 2003, the Minimum Spanning Tree Problem [14] in 2004, and the Partition Problem [24] in 2005.

All these theoretical studies help us to understand how EAs and their variants find a global optimum on specific problems. However, we need more rigorous research on NP-hard problems (which sometimes needs exponential time). A few results studied NP-hard problems but with restriction on the input cases. Such as Storch analyzed the Maximum Clique Problem but only in planar graphs in 2006 [21]; Oliveto, He and Yao analyzed the Vertex Cover Problem, but only focused on Papadimitriou-Steiglitz graphs in 2007 [15], and on bipartite graphs in 2008 [17]; Witt analyzed the Vertex Cover Problem but only on sparse random graphs in 2012 [26]; and Sudholt and Zarges analyzed the Vertex Coloring Problem on bipartite graphs, sparse random graphs and planar graphs in 2010 [23].

This implies that the reason why Evolutionary Algorithms are efficient on NP-hard problems is still underdeveloped. In order to make a step to this goal, we investigate the running time of some EA variants on the Maximum Clique Problem (MCP). Because the MCP is NP-complete, which implies every other problem in NP can be transformed into MCP in polynomial time, our study in this paper is also related to other NP-hard problems.

Our main results are: we propose a new metric based on the expected running time to escape a local optimal. We will show the usefulness of this metric, such as how it dominates the expected running time of finding a maximum clique. This metric tells us whether an algorithm has found a maximum clique, with an overwhelming high probability. Then, based on this metric, we will show our analyzed algorithms are expected to find a maximum clique on planar graphs, bipartite graphs

and sparse random graphs in polynomial time in the number of vertices. Finally, the new metric shows if an algorithm takes an exponential time to find a maximum clique of a graph, it must have been trapped into at least one local optimal which is extremely hard to escape. Furthermore, we will show that our previous proposed (1+1) Adaptive Memetic Algorithm (AMA) with a random permutation local search is expected to escape these (hard to escape) local optimal cliques drastically faster than the well-known basic (1+1) EA. The success of our experimental results not only verifies our theoretical analysis, but also shows that our proposed (1+1) AMA outperforms a state-of-art applied Spacing Memetic Algorithm (SMA) [18]. This indicates the benefit of the adaptive strategy combined with the random permutation local search.

The paper is structured as follows. In Section II, we state the basic (1+1) EA, the Dynamic (1+1) EA, the (1+1) MA, and our (1+1) AMA with two different local search approaches– Random Permutation Local Search (RPLS) and Random Complete Local Search (RCLS). In Section III, we first define our new metric, then analyze the upper bounds of each algorithm to find a maximum clique, and finally show that those algorithms are expected to find a maximum clique on certain families of sparse graphs in polynomial time. In Section IV, we show that if a graph needs an exponential time to escape a local optimal clique, the (1+1) AMA with RPLS will be drastically faster to escape than the basic (1+1) EA. In Section V, experimental results provides a running time comparison among the (1+1) EA [4], the Dynamic (1+1) EA [10], (1+1) MA [22], the (1+1) AMA [2], and the SMA [18] on the Maximum Clique Problem. Our conclusions and future work will be given in Section VI.

II. Algorithm Definitions

In this section we give basic definitions of our algorithms and we begin with the following standard notation that will be used throughout this paper.

1) $f(n) = \omega(g(n)) \leftrightarrow \forall k > 0,\ \exists n_0,\ \forall n > n_0,\ g(n) \cdot k < f(n)$
2) $f(n) = \Omega(g(n)) \leftrightarrow \exists k > 0,\ \exists n_0,\ \forall n > n_0,\ g(n) \cdot k \leq f(n)$
3) $f(n) = o(g(n)) \leftrightarrow \forall \epsilon > 0,\ \exists n_0,\ \forall n > n_0,\ f(n) < g(n) \cdot \epsilon$
4) $f(n) = O(g(n)) \leftrightarrow \exists k > 0,\ \exists n_0,\ \forall n > n_0,\ f(n) \leq g(n) \cdot k$
5) $f(n) = \Theta(g(n)) \leftrightarrow \exists k_1 > 0,\ \exists k_2 > 0,\ \exists n_0,\ \forall n > n_0,\ g(n) \cdot k_1 \leq f(n) \leq g(n) \cdot k_2$
6) $\lim_{n\to\infty} (1 + 1/n)^n = e$

A. The Maximum Clique Problem

A *clique* of a graph is a subset of vertices from this graph such that every two vertices in the subset are connected by an edge. The *Maximum Clique Problem* is the NP-hard problem of finding the largest size of a clique in a graph. In this section, we will formalize a fitness function f_{MCP} for the Maximum Clique Problem.

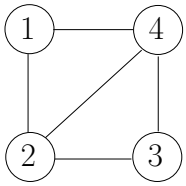
For a given graph $G = (V = \{v_1, v_2, ..., v_n\}, E)$, a bit string $x = (x_1, x_2, ..., x_n) \in \{0,1\}^n$ defines a Maximum Clique potential solution (an induced subgraph) where $x_i = 1$ represents that vertex v_i is selected. We say x represents a clique if each selected vertex in x is connected to all other selected vertices in x, i.e. $\{(v_i, v_j) \mid x_i = x_j = 1 \text{ and } i \neq j\} \subseteq E$.

Definition 1. The fitness function f_{MCP} is defined as follows:

$$f_{\text{MCP}}(x) = \begin{cases} \texttt{ONEMAX}(x), & \text{if x represents a clique,} \\ -\texttt{LackEdges}(x), & \text{otherwise,} \end{cases}$$

where $\texttt{ONEMAX}(x)$ is the number of ones in x; and $\texttt{LackEdges}(x)$ is the number of missing edges such that the subgraph becomes a clique.

Example 2. For a given graph G displayed below, $f_{\text{MCP}}(1101) = 3$ because $x = (1101)$ is a clique consists of vertices 1, 2 and 4. $f_{\text{MCP}}(1111) = -1$ because we need to add at least one edge $(1, 3)$.



A maximum clique for a graph G is a global optimal solution x that maximizes $f_{\text{MCP}}(x)$.

B. Algorithms to be analyzed

The algorithms we will analyze on the Maximum Clique Problem are the (1+1) EA [4], the Dynamic (1+1) EA [10], the (1+1) MA [22] and the (1+1) AMA [2]. Note these algorithms all try to maximize a function f. The time complexity analysis in this paper looks at the number of evaluations of this fitness (objective) function $f = f_{\text{MCP}}$. The algorithms are stated as below:

Algorithm 3. (1+1) EA.

1) $p_m := 1/n$.
2) Choose randomly an initial bit string $x \in \{0,1\}^n$.
3) Repeat the following mutation step:
   a) Compute x' by flipping independently each bit x_i with probability p_m.
   b) If $f(x') \geq f(x)$ then $x := x'$.

Algorithm 4. Dynamic (1+1) EA.

1) Choose a sequence $p_t(n) \in (0, 1/2)$ called mutation probabilities for step t.
2) Choose $x \in \{0,1\}^n$ uniformly at random. $t := 1$.
3) Let y be the result of flipping each bit in x independently with probability $p_t(n)$ (mutation).
4) If $f(y) \geq f(x)$ then $x := y$ (selection).
5) Increase t by 1.
6) Stop if meet some stopping criterion; otherwise, go to step 3.

where $p_t(n) = 2^{t^*}/n$ with $t^* \equiv (t-1) \bmod (\lceil \log n \rceil - 1)$.

Algorithm 5. (1+1) MA.

1) Choose $x \in \{0,1\}^n$ uniformly at random. $x := \texttt{LocalSearch}(x)$.
2) $y := x$. Flip every bit in y with probability p_m. $y := \texttt{LocalSearch}(y)$.
3) If $f(y) \geq f(x)$ then $x := y$.
4) Go to step 2.

where the Local Search in Step 1 and 2 is a Random Complete Local Search which we will state below.

Algorithm 6. (1+1) AMA.

1) Initialize the mutation probability $p \in [0, 1/2]$.
2) Choose $x \in \{0,1\}^n$ uniformly at random.
3) $y := \texttt{Mutation}(x)$.
4) $z := \texttt{LocalSearch}(y)$.
5) $p := \texttt{Adaptive}(f(x), f(z), p)$.
6) If $f(z) \geq f(x)$ then $x := z$.
7) Stop if meet some stopping criterion; otherwise, go to step 3.

where the adaptive function in step 5 is chosen as below:

$$p = \begin{cases} \frac{1}{n}, & \text{if } f(z) > f(x) \text{ or } p = \frac{1}{2} \\ \min\left(2p, \frac{1}{2}\right), & \text{otherwise.} \end{cases}$$

C. Two local searches

As stated before, the local search in MAs can have many variations. We will analyze a *Random Complete Local Search* (RCLS) and a *Random Permutation Local Search* (RPLS) in this paper.

Algorithm 7. Random Complete Local Search (RCLS). For a given string $x \in \{0,1\}^n$:

1) $\texttt{BestNeighborSet} := \left\{ \begin{array}{l} y \mid f(y) > f(x), \texttt{Hamming}(x,y) = 1, \text{ and} \\ \forall z \text{ with } \texttt{Hamming}(x,z) = 1 \rightarrow f(y) \geq f(z) \} \end{array} \right\}$.
2) Stop and return x if $\texttt{BestNeighborSet} = \emptyset$.
3) x is randomly choosen from $\texttt{BestNeighborSet}$.
4) Go to step 1.

where $\texttt{Hamming}(x,y)$ is the number of different bits between x and y. Note that the RCLS will evaluate all n neighbors and then select one flip, thus the RCLS will stop on the Maximum Clique Problem within $2n^2$ fitness evaluations, or there is no neighbor solution which has a better fitness value, i.e. $\texttt{BestNeighborSet} = \emptyset$.

Unlike RCLS that evaluates all n neighbors before selecting one flip, RPLS randomly generates a permutation to represent the sequence of bits to search and executes the flipping as soon as the fitness evaluation improves. The algorithm is stated as below:

Algorithm 8. Randomized Permutation Local Search (RPLS).

For a given string $x = (x_1, x_2, \cdots, x_n) \in \{0,1\}^n$:

1) Generate a random permutation $\texttt{Per}$ of length n.
2) $i := 1$, $\texttt{WorseCount} := 0$.
3) $y := \texttt{flip}(x, \texttt{Per}[i])$.
4) If $f(y) > f(x)$ then $x := y$, $\texttt{WorseCount} := 0$.
5) $\texttt{WorseCount} := \texttt{WorseCount} + 1$.
6) $i := (i \mod n) + 1$.
7) Stop if the stopping criterion holds (see below). Otherwise, go to step 3.

Here $\texttt{flip}(x, \texttt{Per}[i])$ denotes that the $\texttt{Per}[i]$-th bit in x is flipped, and $\texttt{Per}[i]$ is the i-th number in the permutation $\texttt{Per}$.

Note that the RPLS uses only one fitness evaluation per step, so the RPLS will stop on the Maximum Clique Problem within $2n$ fitness evaluations, or there is no neighbor solution which has a better fitness value, i.e. $\texttt{WorseCount} = n$.

Example 9. Suppose string $x = (0,0,0,0)$, and $\texttt{Per} = (3,2,1,4)$. So the RPLS will first check a possible flipping for the third bit in x to get $x' = (0,0,1,0)$. If $f(x') > f(x)$ then $x := x'$. This check sequence follows $\texttt{Per}$ in a cyclic fashion. That is, after checking the fourth bit in x, the PRLS will restart checking the third bit in x.

Therefore, the expected running time for the local search on the Maximum Clique Problem is $O(n^2)$ for RCLS and $O(n)$ for RPLS. In the rest of this paper, we will use AMA_RCLS to denote the algorithm AMA using RCLS as the local search, and use AMA_RPLS to denote the algorithm AMA using RPLS as the local search.

III. New Metric on Stagnation Analysis

Now we analyze how these algorithms are coping with the stagnation when they are trapped into a local optimal clique. We first define a new metric to measure the difficulty of escaping out of a local optimal clique. Then we show how can this new metric dominates the time complexity of each analyzed algorithm to find a maximum clique, so as to show the usefulness of the new metric.

Definition 10. For a given clique $x = (x_1, x_2, \ldots, x_n) \in \{0,1\}^n$ in graph G, function $\texttt{BLOCKONES}(x)$ is formalized as:

$$\texttt{BLOCKONES}(x) = \min_{(y_1,y_2,\ldots,y_n) \in Clique_{>x}} \left(\sum_{i=1}^{n} x_i \overline{y_i} \right),$$

where $Clique_{>x}$ is the set of all cliques in G with clique size greater than the clique size of x, i.e. $f_{\texttt{MCP}}(y) > f_{\texttt{MCP}}(x)$ for all $y \in Clique_{>x}$. Note, the complement of y_i is $\overline{y_i} = 1 - y_i$.

So $\texttt{BLOCKONES}(x)$ is the minimal number, such that at least $\texttt{BLOCKONES}(x)$ number of ones in x are blocking x to find a larger clique in G. So we have $\texttt{BLOCKONES}(x) = 0$ if the clique of x is a subset of a larger clique. Also $0 < \texttt{BLOCKONES}(x) < n/2$ if x is a local optimal clique.

Lemma 11. *If the analyzed algorithms are stagnated at a local optimal solution x, let $t := \texttt{BLOCKONES}(x)$, the expected running time to skip out of this local optimal solution and find a larger clique is bounded by*

1) $O\big(n^{2t+1}\big)$ *for the (1+1) EA,*
2) $O\big(n^{2t+1}\log n\big)$ *for the Dynamic (1+1) EA,*
3) $O\big(n^{2t+2}\big)$ *for the (1+1) MA,*
4) $O\big(n^{2t+2}\log n\big)$ *for the (1+1) AMA_RCLS, and*
5) $O\big(n^{2t+1}\log n\big)$ *for the (1+1) AMA_RPLS.*

Proof. Since $t := \texttt{BLOCKONES}(x)$, there must exist a larger clique y such that $f_{\texttt{MCP}}(y) = f_{\texttt{MCP}}(x) + 1$ and $\sum_{i=1}^{n} \overline{y_i} x_i = t$.

Part 1. The (1+1) EA and the Dynamic (1+1) EA both need to flip those blocking t bits in x from one to zero, and also flip another $t+1$ bits in x from zero to one to find this larger clique y. So the probability of the (1+1) EA and the Dynamic (1+1) EA to find this larger clique in one mutation is: $\texttt{Prob}_{\texttt{success}} = p^{2t+1}(1-p)^{n-2t-1} = \Omega\big(p^{2t+1}e^{-p(n-2t-1)}\big)$, where p is the mutation probability. So if $p = 1/n$, $\texttt{Prob}_{\texttt{success}} = \Omega\big(n^{-(2t+1)}\big)$, the (1+1) EA is expected to skip out of the local optimal x and find a larger clique in $O\big(n^{2t+1}\big)$ steps. And for the Dynamic (1+1) EA, because the dynamic mutation probability has a $p = 1/n$ in every $\lceil \log n \rceil$ mutations, the upper bound of the Dynamic (1+1) EA is proved.

Part 2. Note that the (1+1) MA, the (1+1) AMA_RCLS, and the (1+1) AMA_RPLS both have a local search approach, so these algorithms can flip t bits from one to zero and flip another t bits from zero to one to get a new clique with the same clique size as x, and this new clique is also a sub-clique of y. Then the local search will flip at least one more bit to get a larger clique (note this larger clique may not be y). So the probability that the local search will find a larger clique after this mutation is one. And the probability of this mutation happens is: $\texttt{Prob}_{\texttt{success}} = p^{2t}(1-p)^{n-2t} = \Omega\big(p^{2t}e^{-p(n-2t)}\big)$.

So if $p = 1/n$, $\texttt{Prob}_{\texttt{success}} = \Omega\big(n^{-2t}\big)$. And since the local search RPLS needs $O(n)$ steps after each mutation (see Algorithm 8), and RCLS needs $O\big(n^2\big)$ steps after each mutation (see Algorithm 7); also the (1+1) AMA_RPLS and the the (1+1) AMA_RCLS have at least one mutation with probability $p = 1/n$ in every $\lceil \log n \rceil$ mutations, the rest upper bounds are proved. □

We claim Lemma 11 is important because it indicates that if any of the analyzed algorithm has trapped into a local optimal solution for more than this number of times evaluating the fitness function, the probability that the algorithm can find a larger clique in the future is exponentially small. Note that due to Definition 10, $\texttt{BLOCKONES}(x) \leq \texttt{ONEMAX}(x)$. Thus each running algorithm knows an upper bound of t in Lemma 11. So Lemma 11 can indicate each running algorithm to stop with an overwhelming probability that a maximum clique has been found.

For example, if the (1+1) AMA_RPLS has found a clique of five nodes, then we know $\texttt{BLOCKONES}(x) \leq 5$. So according to Lemma 11, if we could not find a larger clique in the next $O\big(n^{2\cdot 5+1}\log n\big)$ fitness evaluations, we can claim this 5-clique is a maximum clique, and the probability of our claim to be false is exponentially small.

Definition 12. For a given graph G, the function $\texttt{MAXBLOCKONES}\ (G)$ is formalized as:

$$\texttt{MAXBLOCKONES}(G) = \max\{\texttt{BLOCKONES}(x) \mid x \text{ is a clique in G}\}.$$

Theorem 13. *For a given graph G, let* $t := \texttt{MAXBLOCKONES}(G)$. *The expected running time of the analyzed algorithms to find a maximum clique of G is bounded by*

1) $O\big(n^{2t+2}\big)$ *for the (1+1) EA,*
2) $O\big(n^{2t+2}\log n\big)$ *for the Dynamic (1+1) EA,*
3) $O\big(n^{2t+3}\big)$ *for the (1+1) MA,*
4) $O\big(n^{2t+3}\log n\big)$ *for the (1+1) AMA_RCLS, and*
5) $O\big(n^{2t+2}\log n\big)$ *for the (1+1) AMA_RPLS.*

Proof. Part 1. We prove the upper bounds of the (1+1) EA and the Dynamic (1+1) EA in two steps: (a) if the start string x does not represent a clique, then the expected running time of finding a clique is bounded by $O\big(n^2\big)$ and $O\big(n^2\log n\big)$ respectively; and (b) if the start string x represents a clique, then the expected running time of finding a maximum clique is bounded by $O\big(n^{2t+2}\big)$ and $O\big(n^{2t+2}\log n\big)$ respectively.

Step (a): if x does not represent a clique, the function $f_{\texttt{MCP}}$ will guide the algorithm to flip many ones to zeros to find a clique. The probability of the (1+1) EA and the Dynamic (1+1) EA to flip at least one bit in x from one to zero is $\texttt{Prob}_{\texttt{success}} = \Omega\big(p^1(1-p)^{n-1}\big)$. And $\texttt{Prob}_{\texttt{success}} = \Omega(1/n)$ when $p = 1/n$. Note we have one mutation with $p = 1/n$ in every $\lceil \log n \rceil$ mutations for the Dynamic (1+1) EA. So we will expect to flip at least one bit in x from one to zero in $O(n)$ mutations on the (1+1) EA and $O(n\log n)$ mutations on the Dynamic (1+1) EA. Also, x has at most n bits of ones, so we will expect to find a clique in $O\big(n^2\big)$ mutations on the (1+1) EA and $O\big(n^2\log n\big)$ mutations on the Dynamic (1+1) EA.

Step (b): Based on Lemma 11, the (1+1) EA and the Dynamic (1+1) EA can skip out of a local optimal clique and find a larger clique by $O\big(n^{2t+1}\big)$ and $O\big(n^{2t+1}\log n\big)$ mutations respectively. Also, a maximum clique of G will be obtained before this skip is performed n times.

So the (1+1) EA and the Dynamic (1+1) EA is expected to find a maximum clique of G in $O\big(n^{2t+2}\big)$ and $O\big(n^{2t+2}\log n\big)$ mutations (i.e. fitness evaluations) respectively.

Part 2. We prove the upper bounds of the (1+1) MA, the (1+1) AMA_RCLS and the (1+1) AMA_RPLS. The proof is similar to Part 1. We have stated that if the start string x does not represent a clique, the RPLS and the RCLS are expected to find a clique in $O(n)$ and $O\big(n^2\big)$ steps respectively (in Algorithm 8 and Algorithm 7). Also, from Lemma 11, the upper bounds for the three memetic based algorithms to skip out of a local optimal clique and find a larger clique is known. Since each time this skip will increase the clique size by at least one, a maximum clique of G will be obtained before this skip is performed n times. So the upper bounds of the (1+1) MA, the (1+1) AMA_RCLS and the (1+1) AMA_RPLS are proved. □

Corollary 14. *For a graph G, let $t := \texttt{MAXBLOCKONES}(G)$. We have*

1) *If $t = \Theta(1)$, the analyzed algorithms are expected to find a maximum clique of G in a polynomial time.*
2) *If $t = \omega(1)$ and $t = o(n)$, the analyzed algorithms are expected to find a maximum clique of G in a sub-exponential time.*
3) *If any analyzed algorithm has took an exponential time to find a maximum clique on a graph G, this algorithm must have been trapped into at least one local optimal clique x with $\texttt{BLOCKONES}(x) = \Theta(n)$.*

Theorem 13 and Corollary 14 show that

1) For all bipartite graphs and planar graphs, the analyzed algorithms are expected to find a maximum clique within a polynomial time. This is because the maximum clique size is three for planar graphs, and two for bipartite graphs.
2) For all sparse random graphs in the $G(n, c/n)$ model (e.g. edges between n nodes are connected with probability c/n, where $c > 0$ is a constant [26]), the analyzed algorithms are expected to find a maximum clique within a polynomial time. Note a graph in this model is built by inserting all possible edges independently with probability c/n. Thus the expected vertex degree is $c(n-1)/n$ which is $\Theta(1)$. Hence it falls into the category of $t = \Theta(1)$ in Corollary 14.

IV. Ability to avoid stagnation

From Section III, we see that the ability of jumping out of a local optimal clique x with a large $\texttt{BLOCKONES}(x)$ is very important because it is the most time consuming part and dominates the time complexity of finding a maximum clique. This section we will analyze this ability and show that for any local optimal clique x with a very large $\texttt{BLOCKONES}(x)$, our proposed (1+1) AMA_RPLS algorithm is expected to take much less running time than the well-known (1+1) EA with a mutation probability $p = 1/n$ to jump out of x and find a better clique. First we define some formulas that will be used in this section.

Definition 15. To measure the probabilities of jumping from a local optimal clique x to another clique y with $f_{\texttt{MCP}}(y) \geq f_{\texttt{MCP}}(x)$. Let $U_1 := \{i \mid x_i = 1, y_i = 0 \text{ and } 1 \leq i \leq n\}$, and $U_2 := \{i \mid x_i = 0, y_i = 1 \text{ and } 1 \leq i \leq n\}$, then we have (a) $|U_2| \geq |U_1|$, and (b) $|U_1| \geq \texttt{BLOCKONES}(x)$ if $|U_2| > |U_1|$ (due to Definition 10).

Hence to jump from x to y, we need to flip all bits in U_1 from one to zero, and flip all bits in U_2 from zero to one. Now we define $\texttt{Prob}^{\texttt{EA}}_{(x\to y)}$, $K^U_{\texttt{RPLS}}$, $\texttt{Prob}^{+}_{(x\to y)}$ and $\texttt{Prob}^{-}_{(x\to y)}$ as below:

1) Let $\texttt{Prob}^{\texttt{EA}}_{(x\to y)}$ be the probability of the (1+1) EA jumping from x to y using one mutation. Then we have:
$$\texttt{Prob}^{\texttt{EA}}_{(x\to y)} = p^{|U_1|}p^{|U_2|}(1-p)^{n-|U_1|-|U_2|}.$$
2) For a set of bits U, let $K^U_{\texttt{RPLS}}$ be the probability that the RPLS will first check all bits in U according to the permutation array. So the permutation array will have all bits in U prior to the other $(n-|U|)$ bits. Then the probability of getting this type of permutation array is:
$$K^U_{\texttt{RPLS}} = \frac{|U|!(n-|U|)!}{n!} \geq \frac{1}{(n-|U|+1)^{|U|}}$$
3) Let $\texttt{Prob}^{+}_{(x\to y)}$ be the probability of the (1+1) AMA_RPLS jumping from x to y using one mutation with restriction that
   a) The mutation flips all bits in U_1 from one to zero, and keeps $n-|U_1|-|U_2|$ bits not been flipped. Note none of these $n-|U_1|-|U_2|$ bits is in U_1 or U_2. Thus the mutation reaches a sub-clique of both x and y.
   b) The local search RPLS first checks all bits in U_2 and flips them if they are not ones.

   Then we have:
$$\texttt{Prob}^{+}_{(x\to y)} = p^{|U_1|}(1-p)^{n-|U_1|-|U_2|}K^{U_2}_{\texttt{RPLS}},$$
   where $K^{U_2}_{\texttt{RPLS}}$ is the probability that the RPLS will first check all bits in U_2 according to the permutation array.
4) Let $\texttt{Prob}^{-}_{(x\to y)}$ be the probability of the (1+1) AMA_RPLS jumping from x to y using one mutation with restriction that
   a) The mutation flips all bits in U_2 from zero to one, and keeps $n-|U_1|-|U_2|$ bits not been flipped. Note none of these $n-|U_1|-|U_2|$ bits is in U_1 or U_2. Thus the mutation reaches a bit string which does not represent a clique.
   b) The local search RPLS first checks all bits in U_1 and flips them if they are not zeros, i.e. the RPLS reaches y by the $-\texttt{LackEdges}$ part in the fitness function (see Definition 1).

   Then we have:
$$\texttt{Prob}^{-}_{(x\to y)} = p^{|U_2|}(1-p)^{n-|U_1|-|U_2|}K^{U_1}_{\texttt{RPLS}},$$
   where $K^{U_1}_{\texttt{RPLS}}$ is the probability that the RPLS will first check all bits in U_1 according to the permutation array.

Theorem 16. *Let x and y be two cliques with $f_{\texttt{MCP}}(y) \geq f_{\texttt{MCP}}(x)$. Let $U_1 := \{i \mid x_i = 1, y_i = 0$ and $1 \leq i \leq n\}$. If $|U_1| = \Theta(n)$, then the expected running time of the (1+1) AMA_RPLS directly jumping from x to y is exponentially faster than the expected running time of the (1+1) EA with a mutation probability $p = 1/n$ directly jumping from x to y.*

Note "directly jumping" denotes the algorithm only use one mutation to reach the destination y. This is to distinguish with the algorithm using multiple mutations where each mutation jumps to another clique and finally reaches the destination y.

Proof. Let $U_2 := \{i \mid x_i = 0, y_i = 1 \text{ and } 1 \leq i \leq n\}$. Since $f_{\texttt{MCP}}(y) \geq f_{\texttt{MCP}}(x)$ and $|U_1| = \Theta(n)$, we have $|U_2| \geq |U_1| = \Theta(n)$.

Then the success probability of the (1+1) EA with $p = 1/n$ to find y in one mutation is: $\texttt{Prob}^{\text{EA}}_{x\to y} = p^{|U_1|}p^{|U_2|}(1-p)^{n-|U_1|-|U_2|} = (\frac{1}{n})^{|U_1|+|U_2|}(1-\frac{1}{n})^{n-|U_1|-|U_2|} = (\frac{1}{n})^{\Theta(n)}$.

The success probability of the (1+1) AMA_RPLS with $p = 1/2$ to find y in one mutation is: $\texttt{Prob}^{\text{AMA\_RPLS}}_{x\to y} > p^{|U_1|}p^{|U_2|}(1-p)^{n-|U_1|-|U_2|} = (\frac{1}{2})^{|U_1|+|U_2|}(1-\frac{1}{2})^{n-|U_1|-|U_2|} = (\frac{1}{2})^n$.

Thus the success probability of the (1+1) AMA_RPLS with $p = 1/2$ directly mutating from x to y is exponentially larger than the success probability of the (1+1) EA with $p = 1/n$ directly mutating from x to y.

Recall that the (1+1) AMA_RPLS has a dynamic mutation approach and a local search, so it will have at least one mutation with $p = 1/2$ in every $\log n$ mutations, and every mutation is followed by $O(n)$ steps of local search. But an exponential large number divided by a polynomial large number is still exponential. Thus the expected running time of the (1+1) AMA_RPLS directly jumping from x to y is still exponentially faster than the expected running time of the (1+1) EA directly jumping from x to y. □

Lemma 17. *In Definition 15, if $|U_1| = \omega(1)$, then the probabilities $\texttt{Prob}^{+}_{(x\to y)}$ and $\texttt{Prob}^{-}_{(x\to y)}$ with mutation $p = \Theta(|U_1|/n)$ are super-polynomially larger than the same probabilities with mutation $p = 1/n$ respectively. I.e. both ratios of $\frac{p_1^{|U_1|}(1-p_1)^{n-|U_1|-|U_2|}K^{U_2}_{RPLS}}{p_2^{|U_1|}(1-p_2)^{n-|U_1|-|U_2|}K^{U_2}_{RPLS}}$ and $\frac{p_1^{|U_2|}(1-p_1)^{n-|U_1|-|U_2|}K^{U_1}_{RPLS}}{p_2^{|U_2|}(1-p_2)^{n-|U_1|-|U_2|}K^{U_1}_{RPLS}}$ are super-polynomially large when $p_1 = \Theta(|U_1|/n)$ and $p_2 = 1/n$.*

Proof. Let $p_1 = \Theta(|U_1|/n)$ and $p_2 = 1/n$. Then for the probability $\texttt{Prob}^{+}_{(x\to y)}$, the ratio of p_1 and p_2 is:

$$\frac{p_1^{|U_1|}(1-p_1)^{n-|U_1|-|U_2|}}{p_2^{|U_1|}(1-p_2)^{n-|U_1|-|U_2|}} = \left(\frac{p_1}{p_2}\right)^{|U_1|}\left(\frac{1-p_1}{1-p_2}\right)^{n-|U_1|-|U_2|},$$

We have $\left(\frac{1-p_1}{1-p_2}\right)^{n-|U_1|-|U_2|} > (1-p_1)^{n-|U_1|-|U_2|} > (1-p_1)^n$ and $(1-p_1)^n = \Theta(e^{-|U_1|})$, since $p_1 = \Theta(|U_1|/n)$. And because $p_1 = \Theta(|U_1|/n)$, $p_2 = 1/n$, we have $\left(\frac{p_1}{p_2}\right)^{|U_1|} = (\omega(1))^{|U_1|}$. Thus this ratio is dominated by $\left(\frac{p_1}{p_2}\right)^{|U_1|}$. Also, because $|U_1| = \omega(1)$, this ratio is super-polynomially large.

Because $|U_2| \geq |U_1|$, the proof for the probability $\texttt{Prob}^{-}_{(x\to y)}$ is the same as above. □

For the following theorem, recall that "directly jumping" denotes the algorithm only use one mutation to reach the destination y.

Theorem 18. *Let x and y be two cliques with $f_{\text{MCP}}(y) \geq f_{\text{MCP}}(x)$. Let $U_1 := \{i \mid x_i = 1, y_i = 0 \text{ and } 1 \leq i \leq n\}$. If $|U_1| = \omega(1)$ and $|U_1| = o(n)$, then the expected running time of the (1+1) AMA_RPLS directly jumping from x to y is super-polynomially faster than the expected running time of the (1+1) EA with a mutation probability $p = 1/n$ directly jumping from x to y.*

Proof. Let $U_2 := \{i \mid x_i = 0, y_i = 1 \text{ and } 1 \leq i \leq n\}$. Then we have $|U_2| \geq |U_1| = \omega(1)$ since $f_{\text{MCP}}(y) \geq f_{\text{MCP}}(x)$ and $|U_1| = \omega(1)$. According to Definition 15, the ratio of $\texttt{Prob}^{+}_{(x\to y)}$ and $\texttt{Prob}^{\text{EA}}_{(x\to y)}$ is:

$$\frac{\texttt{Prob}^{+}_{(x\to y)}}{\texttt{Prob}^{\text{EA}}_{(x\to y)}} = \frac{1}{(n-|U_2|+1)^{|U_2|}}\frac{1}{p^{|U_2|}} > 1.$$

We have this ratio is greater than one when $p = 1/n$. Note this is the probability ratio of the (1+1) AMA_RPLS with mutation probability $p = 1/n$ and the (1+1) EA with mutation probability $p = 1/n$.

In the (1+1) AMA_RPLS, the dynamic mutation approach obtains a mutation probability between $|U_1|/n$ and $2|U_1|/n$ in every $\log n$ mutations unless it finds a larger clique earlier. And according to Lemma 17, since $|U_1| = \omega(1)$, the ratio of the $\texttt{Prob}^{+}_{(x\to y)}$ with a mutation probability between $|U_1|/n$ and $2|U_1|/n$ and the $\texttt{Prob}^{+}_{(x\to y)}$ with a mutation probability $p = 1/n$ is super-polynomially large.

So the probability of the (1+1) AMA_RPLS directly jumping from x to y is super-polynomially larger than the probability of the (1+1) EA directly jumping from x to y. □

Theorem 19. *For a local optimal clique x with $t := \texttt{BLOCKONES}(x)$, we have if $t = \Theta(n)$, the (1+1) AMA_RPLS is expected to skip out of x and find a larger clique super-polynomially faster than the (1+1) EA with a mutation probability $p = 1/n$.*

Proof. To prove the theorem, we just need to show that if both algorithms are trapped into x, then for any clique y such that y is the first clique the algorithms have found with $f_{\text{MCP}}(y) > f_{\text{MCP}}(x)$ (escape from x), the probability of the (1+1) AMA_RPLS jumping from x to y is super-polynomially larger than the probability of the (1+1) EA with $p = 1/n$ jumping from x to y.

Moreover, since the (1+1) EA only accepts a new solution if its fitness is greater or equal to the current solution, it can only escape from clique x to clique y by two ways: (a) directly mutating from x to y; or (b) mutating multiple times to different cliques with the same clique size as x, and then mutating to the clique y.

Proof of case (a), by directly mutating from x to y.
Let $U_1 := \{i \mid x_i = 1, y_i = 0 \text{ and } 1 \leq i \leq n\}$. Since $f_{\text{MCP}}(y) > f_{\text{MCP}}(x)$, we have $|U_1| \geq t = \Theta(n)$.

Because $|U_1| = \Theta(n)$, from Theorem 16 we know the probability ratio of the (1+1) AMA_RPLS directly jumping from x to y and the (1+1) EA directly jumping from x to y is exponentially large.

Proof of case (b), by mutating multiple times to different cliques with the same clique size as x, and then mutating to the clique y.

Let $\texttt{Path}_\lambda$ be an arbitrary $(\lambda+1)$-length path $x^0 \to x^1 \to x^2 \cdots x^\lambda \to y$, where $x^0 = x$, $\lambda \geq 0$ and each x^i is a bit string with $f_{\text{MCP}}(x) = f_{\text{MCP}}(x^i)$. We proof this part by proving that the probability of the (1+1) AMA_RPLS jumping along this $\texttt{Path}_\lambda$ to reach y is super-polynomially larger than the

probability of the (1+1) EA jumping along this $\texttt{Path}_\lambda$ to reach y when $t = \Theta(n)$. The proof contains four steps.

Step 1. For any jump from x^i to x^{i+1} ($0 \leq i \leq \lambda - 1$) in $\texttt{Path}_\lambda$, let x^i_j denote the j-th bit of the bit string x^i, and let $\mathcal{U}_i = \{j \mid x^i_j = 1, x^{i+1}_j = 0 \text{ and } 1 \leq j \leq n\}$. So we will flip $|\mathcal{U}_i|$ number of bits from one to zero and flip another $|\mathcal{U}_i|$ number of bits from zero to one. So according to Definition 15, we know $\texttt{Prob}^{-}_{(x^i \to x^{i+1})} = \texttt{Prob}^{+}_{(x^i \to x^{i+1})} > \texttt{Prob}^{\texttt{EA}}_{(x^i \to x^{i+1})}$. Thus we have:

$$\begin{aligned} \frac{\texttt{Prob}^{\texttt{AMA\_RPLS}}_{(x^i \to x^{i+1})}}{\texttt{Prob}^{\texttt{EA}}_{(x^i \to x^{i+1})}} &> \frac{\texttt{Prob}^{+}_{(x^i \to x^{i+1})} + \texttt{Prob}^{-}_{(x^i \to x^{i+1})}}{\texttt{Prob}^{\texttt{EA}}_{(x^i \to x^{i+1})}} \\ &= \frac{2}{(n - |\mathcal{U}_i| + 1)^{|\mathcal{U}_i|}} \frac{1}{p^{|\mathcal{U}_i|}} > 2. \end{aligned} \tag{1}$$

Furthermore, this ratio is exponentially large if $|\mathcal{U}_i| = \Theta(n)$ (in Theorem 16), or this ratio is super-polynomially large if $|\mathcal{U}_i| = \omega(1)$ and $|\mathcal{U}_i| = o(n)$ (in Theorem 18).

Step 2. Also according to Definition 15, for the last jump from x^λ to y, we have:

$$\frac{\texttt{Prob}^{\texttt{AMA\_RPLS}}_{(x^\lambda \to y)}}{\texttt{Prob}^{\texttt{EA}}_{(x^\lambda \to y)}} > \frac{\texttt{Prob}^{+}_{(x^\lambda \to y)}}{\texttt{Prob}^{\texttt{EA}}_{(x^\lambda \to y)}} > 1,$$

and this ratio is exponentially large if this jump needs to flip $\Theta(n)$ bits (in Theorem 16), or this ratio is super-polynomially large if the number of bits flipped is between $\omega(1)$ and $o(n)$ (in Theorem 18).

Step 3. Let $\texttt{Prob}^{\texttt{AMA\_RPLS}}_{\texttt{success}}$ and $\texttt{Prob}^{\texttt{EA}}_{\texttt{success}}$ be the probabilities of the (1+1) AMA_RPLS and the (1+1) EA with $p = 1/n$ jumping along this itinerary to reach y, respectively. We have:

$$\frac{\texttt{Prob}^{\texttt{AMA\_RPLS}}_{\texttt{success}}}{\texttt{Prob}^{\texttt{EA}}_{\texttt{success}}} = \prod_{i=0}^{\lambda-1} \frac{\texttt{Prob}^{\texttt{AMA\_RPLS}}_{(x^i \to x^{i+1})}}{\texttt{Prob}^{\texttt{EA}}_{(x^i \to x^{i+1})}} \frac{\texttt{Prob}^{\texttt{AMA\_RPLS}}_{(x^\lambda \to y)}}{\texttt{Prob}^{\texttt{EA}}_{(x^\lambda \to y)}}.$$

Step 4. Recall that $t = \texttt{BLOCKONES}(x)$, and $f_{\texttt{MCP}}(y) > f_{\texttt{MCP}}(x)$, thus to jump along this itinerary from x to y, we will finally flip at least t bits from one to zero. And since $t = \Theta(n)$, to achieve our final goal of finding y, we have three ways:

1) at least one jump (either belonging to Step 1 or Step 2) needs to flip $\Theta(t)$ bits, and the ratio of this jump is exponentially large (Theorem 16), while the ratios of other jumps are all greater than one, thus the overall ratio in Step 3 is exponentially large; or
2) have at least $\omega(1)$ number of jumps, which are belonging to Step 1, where each jump needs to flip $\omega(1)$ bits. Since the ratio of each jump with flipping $\omega(1)$ bits is super-polynomially large, thus the overall ratio in Step 3 is super-polynomially large; or
3) have $\Omega(t)$ number of jumps, which are belonging to Step 1, where each jump only needs to flip $\Theta(1)$ bits. Since the ratio of these small jumps is greater than two (in Step 1), the overall ratio in Step 3 is still exponentially large.

□

V. Experimental Results

In this section we will test our analyzed algorithms on the Maximum Clique Problem. To avoid only comparing algorithms that are analyzed in theory, we also bring a state-of-art Spacing Memetic Algorithm (SMA) [18] into comparison.

In short, the SMA first keeps the minimum distance between each two individuals above a threshold, and then try to maximize the average distance among the population individuals. Also, it uses an elitist selection approach based on both distance and fitness. Thus, it follows the principle "diversity without quality sacrifices".

Table I reports maximum clique results on some DIMACS instances [11]. Column 1 depicts the graph names with their best known clique size in the parentheses. We test the (1+1) EA [4] in column 2, the Dynamic (1+1) EA [10] in column 3, the (1+1) MA [22] in column 4, our (1+1) AMA_RCLS [2] in column 5, our (1+1) AMA_RPLS [2] in column 6 and the SMA [18] in column 7. Each algorithm is run on each graph 10 times where each run is limited to one minute of running time on a linux machine with 2.5GHz Intel CPU. The sub-column "Best" is the best clique found in 10 runs, and the sub-column "Avg" is the average clique size in 10 runs. The sub-column "Gen" represents the average generations, i.e. average number of iterations.

The "Best" entry is grey-colored if it finds the best known clique of that graph. The "Avg" entry is grey-colored if it has the best average result.

Note we restrict the running time to one minute because it is enough for our proposed (1+1) AMA_RPLS to find a maximum clique in all 10 runs on those small order graphs (such as `brock200_2`, `C125.9`, `gen200_p0.9_55`, etc.), but not all tested algorithms can achieve this within one minute. Meanwhile, even though no algorithm can find a maximum clique on some large order graphs (`C4000.5`, `p_hat1500_3`, etc.) in one minute, our proposed (1+1) AMA_RPLS still outperforms other tested algorithms in terms of the "best" and "Avg" performance.

From Table I, we claim the following:

1) All tested algorithms have found a global optimum in some small order graphs such as `C125.9` and `keller4`. This means that each algorithm found the global optimum if it has enough time.
2) The (1+1) AMA_RPLS outperforms the other five algorithms in most graphs in terms of getting the best "Avg" results. Meanwhile, the "Best" results of the (1+1) AMA_RPLS are greater or equal to the "Best" results in other three algorithms. This denotes that the (1+1) AMA_RPLS has excellent stability.
3) Apart from the population-based SMA, each iteration of the (1+1) EA uses the least running time while each iteration of the (1+1) AMA_RCLS uses the most running time, i.e. the (1+1) EA has the largest value in the "Gen" sub-column for each graph, while the (1+1) AMA_RCLS has the smallest value. This denotes that the local search approaches, especially the RCLS, are very time consuming.
4) The (1+1) AMA_RPLS is the most efficient algorithm that can quickly detect a clique. This can

TABLE I. A COMPARISON OF THE ALGORITHMS ON THE MAXIMUM CLIQUE PROBLEM FOR ONE MINUTE OF CPU TIME (K=10^3, M=10^6).

	(1+1) EA			Dynamic (1+1) EA			(1+1) MA			(1+1) AMA_RCLS			(1+1) AMA_RPLS			(1+1) SMA		
Metrics	Best	Avg	Gen	Best	Avg	Gen	Best	Avg	Gen	Best	Avg	Gen	Best	Avg	Gen	Best	Avg	Gen
brock200_2 (=12)	11	9.8	15.1m	10	9.5	5.4m	11	10.1	170.0k	11	10.9	1.6k	12	12	59.8k	12	11	2.1k
brock200_4 (=17)	16	14.6	14.5m	15	14	5.5m	16	14.9	139.5k	16	15.8	1.7k	17	16.8	57.3k	16	16	2.2k
brock400_2 (=29)	23	22.3	7.6m	22	20.6	2.2m	24	22.9	37.8k	24	22.7	170.0	25	24	11.7k	23	21.9	21.7
brock400_4 (=33)	22	21.2	7.7m	22	19.5	2.2m	24	22.4	38.3k	24	22.7	169.7	24	23.3	11.9k	22	20.9	6.2
brock800_2 (=24)	18	16.5	4.2m	17	15.1	830.4k	18	16.6	3.4k	17	15.4	11.1	19	18.6	2.3k	16	15	1.0
brock800_4 (=26)	15	14.1	4.3m	14	12.4	1.3m	16	15.1	8.5k	16	13.8	16.1	16	16	3.9k	15	13.5	1.5
C125.9 (≥34)	34	33.5	16.2m	34	33.8	10.8m	34	33.5	143.5k	34	34	13.1k	34	34	115.3k	34	34	3.4k
C250.9 (≥44)	44	41.9	9.5m	42	41.2	5.1m	44	42.2	45.5k	44	42.5	1.3k	44	43.4	30.9k	44	42.5	622.2
C500.9 (≥57)	48	44.8	5.6m	45	41.1	2.5m	49	46.4	15.5k	46	45.1	141.4	49	47.9	9.3k	47	45.4	70.3
C1000.9 (≥68)	52	50.2	3.1m	49	45.4	919.7k	53	50.4	890.8	52	49.2	17.6	59	56.2	1.8k	50	46.4	1.1
C2000.5 (≥16)	10	9.5	1.8m	10	9.1	384.7k	fail	fail	0	fail	fail	0	13	11.5	487.5	fail	fail	0
C2000.9 (≥77)	46	41.3	1.7m	42	39	423.6k	fail	fail	0	fail	fail	0	52	49.5	492.8	fail	fail	0
C4000.5 (≥18)	12	9.8	710.1k	fail	fail	70.7k	fail	fail	0	fail	fail	0	13	11.7	81.7	fail	fail	0
DSJC500.5 (≥13)	12	11.3	6.6m	12	10.1	1.5m	12	11.3	31.9k	11	10.9	59.9	13	12.5	6.7k	12	10.8	2.0
DSJC1000.5 (≥15)	12	11.3	3.4m	11	10.3	492.3k	fail	fail	0	fail	fail	0	13	13	1.1	fail	fail	0
gen200_p0.9_44 (=44)	44	40	11.1m	44	39	6.3m	44	39.9	64.8k	44	40.7	2.6k	44	39.9	47.1k	44	39.4	1.1k
gen200_p0.9_55 (=55)	55	46.2	10.5m	55	44.1	6.2m	55	44.7	58.2k	55	55	2.3k	55	55	36.9k	55	50	879.1
gen400_p0.9_55 (=55)	49	47.6	6.5m	47	44.5	2.7m	51	49.1	20.6k	50	48.4	270.8	52	50.2	11.5k	49	47.9	180.0
gen400_p0.9_65 (=65)	48	46.3	6.6m	45	43.7	2.7m	55	48.7	21.1k	49	46	253.7	55	49.3	12.0k	49	46.4	184.7
gen400_p0.9_75 (=75)	55	49.7	6.4m	56	48.8	2.7m	75	65.3	16.1k	75	64.8	287.0	75	65.7	9.9k	75	55.3	161.9
hamming8-4 (=16)	16	13.2	11.6m	16	13.2	4.4m	16	16	90.7k	16	16	850.8	16	16	38.3k	16	16	885
hamming10-4 (=40)	35	33.5	3.2m	30	27.7	720.7k	32	8 fails	20	32	9 fails	2	40	36.3	1.5k	fail	fail	0
keller4 (=11)	11	10.8	17.0m	11	10.3	7.0m	11	11	209.1k	11	11	3.2k	11	11	94.0k	11	11	4.0k
keller5 (=27)	18	17.0	4.4m	18	16.2	1.4m	19	17.8	10.0k	17	16.8	24.1	19	18.9	4.0k	17	16.0	1.2
keller6 (≥59)	26	22.1	933.9k	fail	fail	1.3m	fail	fail	0	fail	fail	0	28	26.4	120.7	fail	fail	0
MANN_a27 (=126)	124	122.4	3.5m	123	120.6	2.7m	125	124.3	6.8k	124	122.8	354.1	126	125.3	5.7k	125	121.9	680.3
MANN_a45 (=345)	331	330.5	673.1k	331	330.1	502k	331	330.4	196.6	332	330.6	10.5	342	339	142.8	331	330.4	1.5
MANN_a81 (≥1100)	364	360.7	403.3k	fail	fail	135.1k	fail	fail	0	fail	fail	0	365	364.6	65.4	fail	fail	0
p_hat300-1 (=8)	8	7.2	10.6m	8	6.8	3.2m	8	7.6	97.3k	8	8	336.8	8	8	25.0k	8	7.8	468.0
p_hat300-2 (=25)	25	24.1	9.5m	25	23.9	2.6m	25	24.6	54.7k	25	25	320.8	25	25	18.1k	25	24.9	373.6
p_hat300-3 (=36)	36	34.5	8.8m	34	32.1	2.9m	36	35.3	41.7k	36	35.4	413.2	36	35.5	18.0k	36	35.4	409.3
p_hat700-1 (=11)	8	7.7	4.9m	8	7.1	951.2k	9	8.5	12.3k	9	8.3	14.6	11	9.8	3.0k	11	8.3	1.5
p_hat700-2 (≥44)	37	35.0	4.5m	30	26	951.0k	38	37.1	6.9k	37	34.8	22.5	38	37.7	2.8k	35	31.9	1.5
p_hat700-3 (≥62)	43	41.0	4.4m	40	33.8	1.4m	43	42.3	8.2k	42	40.8	37.4	43	42.9	4.3k	42	37.4	1.6
p_hat1500-1 (≥12)	9	8.0	2.3m	8	6.9	376.4k	fail	fail	0	fail	fail	0	11	9.4	581.8	fail	fail	0
p_hat1500-2 (≥65)	52	46.2	2.1m	54	44.4	213.9k	fail	fail	0	fail	fail	0	65	62.2	323.8	fail	fail	0
p_hat1500-3 (≥94)	73	67.8	2.0m	63	57.7	269.3k	fail	fail	0	fail	fail	0	88	85.3	390.8	fail	fail	0

be observed from some large order graphs such as `C4000.5`. We claim this is important because there are many real-world problems that do not require the global optimal solutions, but they have strict requirements on the running time.

VI. Conclusions and Future Work

In this paper, we studied several variants of EAs on the Maximum Clique Problem. We proposed a new metric of analyzing the expected running time to escape a local optimal solution. And showed how this metric dominates the expected running time of finding a maximum clique. We also showed, with high probablity, this metric indicates the algorithm will stop with a maximum clique. Then, based on this metric, we showed our analyzed algorithms are expected to find a maximum clique on planar graphs, bipartite graphs and sparse random graphs in polynomial time in the number of vertices. Finally, we showed if an algorithm takes an exponential time to find a maximum clique of a graph, it must have been trapped into at least one local optimal which is extremely hard to escape. Furthermore, we showed that our previous proposed (1+1) Adaptive Memetic Algorithm (AMA) with a random permutation local search is expected to escape these (hard to escape) local optimal cliques drastically faster than the well-known basic (1+1) EA. The success of our experimental results not only verified our theoretical analysis, but also showed that our proposed (1+1) AMA outperforms a state-of-art applied Spacing Memetic Algorithm (SMA) [18]. This indicates the benefit of the adaptive strategy combined with the random permutation local search.

A next step to investigate in the future is to compare the random permutation local search with other local searches on the clique problems with different fitness functions. Also, we would like to try our proposed complexity metric and new local search techniques on other optimization problems.

Acknowledgements

We wish to thank Ralph Versteegen for helpful discussions that improved the content of this paper.

References

[1] E. K. Burke and D. J. L. Silva, "The design of memetic algorithms for scheduling and timetabling problems," in *Recent Advances in Memetic Algorithms, Studies in Fuzziness and Soft Computing*, W. H. N. Krasnogor and J. Smith, Eds., 2004, vol. 166, pp. 289–312.

[2] M. J. Dinneen and K. Wei, "On the analysis of a (1+1) adjusting memetic algorithm," in *Proceedings of Memetic Computing, MC2013*. IEEE, 2013, pp. 24–31.

[3] S. Droste, T. Jansen, and I. Wegener, "On the optimization of unimodal functions with the (1+1) evolutionary algorithm," in *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*. Springer-Verlag, London, UK, 1998, pp. 13–22.

[4] ——, "On the analysis of the (1+1) evolutionary algorithm," *Theoretical Computer Science*, vol. 276, pp. 51–81, 2002.

[5] G. Durrett, F. Neumann, and U. M. O'Reilly, "Computational complexity analysis of simple genetic programming on two problems modeling isolated program semantics," in *Proceedings of the 11th workshop proceedings on Foundations of genetic algorithms. ACM, New York, NY, USA*, 2011, pp. 69–80.

[6] O. Giel and I. Wegener, "Evolutionary algorithms and the maximum matching problem," in *STACS'03: Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science*. Springer-Verlag, London, UK, 2003, pp. 415–426.

[7] T. Jansen, K. A. D. Jong, and I. Wegner, "On the choice of the offspring population size in evolutionary algorithms," *Evol, Comput.*, vol. 13, no. 4, pp. 413–440, 2005.

[8] T. Jansen and I. Wegener, "Evolutionary algorithms: How to cope with plateaus of constant fitness and when to reject strings of the same fitness," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 6, pp. 589–599, 2001.

[9] ——, "On the analysis of evolutionary algorithms—a proof that crossover really can help," *Algorithmica*, vol. 34, no. 1, pp. 47–66, 2002.

[10] ——, "On the analysis of a dynamic evolutionary algorithm," *Journal of Discrete Algorithms*, vol. 4, no. 1, pp. 181–199, 2006.

[11] D. Johnson and M. Trick, "Cliques, coloring and satisfiability second DIMACS implementation challenge, volume 26 of DIMACS series in Discrete Mathematics and Theoretical Computer Science," 1996.

[12] T. Kotzing, D. Sudholt, and M. Theile, "How crossover helps in pseudo-boolean optimization," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation (GECCO '11)*, N. Krasnogor, Ed. ACM, New York, NY, USA, 2011, pp. 989–996.

[13] F. Neri, C. Cotta, and P. Moscato, *Handbook of Memetic Algorithms*. Studies in Computational Intelligence, 2011, vol. 379.

[14] F. Neumann and I. Wegener, "Randomized local search, evolutionary algorithms and the minimum spanning tree problem," in *Proceedings of the annual conference on Genetic and evolutionary computation (GECCO'04)*, 2004, pp. 713–724, lNCS 3102. Springer.

[15] P. S. Oliveto, J. He, and X. Yao, "Evolutionary algorithms and the vertex cover problem," in *Proc. CEC, Singapore*. IEEE, 2007, pp. 1870–1877.

[16] ——, "Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results," *Int'l Journal of Automation and Computing*, vol. 4, no. 3, pp. 281–293, 2007.

[17] ——, "Analysis of population-based evolutionary algorithms for the vertex cover problem," in *Proc. CEC, Hong Kong, China*. IEEE, 2008, pp. 1563–1570.

[18] D. C. Porumbel, J. K. Hao, and P. Kuntz, "Spacing memetic algorithms," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation (GECCO '11), ACM, New York, NY, USA*, N. Krasnogor, Ed., 2011, pp. 1061–1068.

[19] C. Qian, Y. Yu, and Z. H. Zhou, "An analysis on recombination in multi-objective evolutionary optimization," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation (GECCO '11)*, N. Krasnogor, Ed. ACM, New York, NY, USA, 2011, pp. 2051–2058.

[20] G. Rudolph, "Finite markov chain results in evolutionary computation: A tour d'horizon," *Fundamenta Informaticae*, vol. 35, no. 1–4, pp. 67–89, 1998.

[21] T. Storch, "How randomized search heuristics find maximum clique in planar graphs," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation (GECCO '06)*. ACM, New York, NY, USA, 2006, pp. 567–574.

[22] D. Sudholt, "On the analysis of the (1+1) memetic algorithm," in *Proceedings of the 8th annual conference on Genetic and evolutionary computation.*, 2006, pp. 493–500.

[23] D. Sudholt and C. Zarges, "Analysis of an iterated local search algorithm for vertex coloring," in *Proceedings of the 21st International Symposium on Algorithms and Computation (ISAAC)*, ser. LNCS, vol. 6506. Springer, 2010, pp. 340–352.

[24] C. Witt, "Worst-case and average-case aximations by simple randomized search heuristic," in *Proc. of the 22nd Annual Symposium on Theoretical Aspects of Computer Science (STACS'05)*, ser. LNCS, vol. 3804. Springer, 2005, pp. 44–56.

[25] ——, "Runtime analysis of the (μ+1) ea on simple pseudo-boolean functions," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation, Seattle, Washington, USA*, 2006, pp. 651–658.

[26] ——, "Analysis of an iterated local search algorithm for vertex cover in sparse random graphs," *Theoretical Computer Science*, vol. 425, no. 1, pp. 417–425, 2012.