2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Hybridizing the Dynamic Mutation Approach with Local Searches to Overcome Local Optima

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

***Abstract*—A Memetic Algorithm is an Evolutionary Algorithm augmented with local searches. The dynamic mutation approach has been studied extensively in experiments of Memetic Algorithms, but only a few studies in theory. We previously defined a metric BLOCKONES to estimate the difficulty of escaping from a local optima, and showed that the algorithm's ability of escaping from a local optima, that has a large BLOCKONES, is very important, because it dominates the time complexity of finding a global optimal solution. In this paper, we will use the same metric and show the benefits of hybridizing the dynamic mutation approach with one of two local searches, best-improvement and first-improvement. In short, this hybridization greatly enhances the algorithm's ability to escape from any local optima.**

***Keywords*—memetic algorithms, clique problem, runtime analysis**

I. Introduction

A Memetic Algorithm (MA) is a population-based meta-heuristic algorithm that hybridizes Evolutionary Algorithms (EAs) with local searches. This hybridization preserves both the exploratory search ability of evolutionary algorithms and the neighborhood search ability of local searches [1], which has increased interests in MAs. An overview in 2011 shows the usefulness of MAs in many applications [11]. The highlighted experimental results have verified the advantages of MAs, and also motivate the desire for a better understanding of MAs by using runtime analysis.

The first runtime analysis on MAs is the (1+1) MA analyzed by Sudholt [15], in which he compared the (1+1) MA with the (1+1) EA [5] and the randomized local search. The term (1+1) represents that, a) the population size of parents and children are both one, and b) an elitist selection is used, where the next generation will be chosen from both parents and children. Since MAs combine EAs and local searches, it is generally believed that many theoretical findings of EAs are also capable with MAs, for example, some studies show that the population-based EAs are useful [20], [12], [6], [4], and the crossover operation is essential [7], [9], [13]. There are also some theoretical studies that are specific for MAs. For example, Sudholt, in 2006 [14] and in 2009 [16], showed that changing the depth of local searches or the frequency of applying local searches in MAs will reduce the performance from polynomial to super-polynomial. In 2014, Wei and Dinneen in [19] showed that the best local search in MAs for solving the Clique Problem is instance specific; also in [18], they showed that the best fitness function for MAs to solve the Clique Problem is instance specific. Furthermore, the interaction of mutations and local searches has attracted much attention, which is also the focus of this paper. For example, Sudholt and Zarges [17] analyzed the interaction of two different mutations with a local search for Vertex Coloring in 2010; also, Dinneen and Wei [3], in 2013, analyzed a dynamic mutation with two different local searches on some artificially created functions.

The mutation probability is also known as the mutation strength or the mutation step size. There are a great amount of experimental studies show the success of using the dynamic, adaptive, and self-adaptive mutation approaches, see a survey in [10] for more detail. However, only a few runtime analysis studied the dynamic mutation approach. For example, Jansen and Wegener in 2006 [8], and Dinneen and Wei in 2013 [3] studied the dynamic mutation approach on EAs and MAs respectively. They both showed that the dynamic mutation approach can outperform each of static mutation approaches on some artificially created functions, but there also exists some other functions that a static mutation approach outperforms the dynamic mutation approach. We believe that the dynamic mutation approach contributes far more in experiments than the above runtime analyses show us. In our previous study [2], we defined a new metric, called BLOCKONES, to estimate the difficulty of escaping from a local optima and finding a better solution to the Clique Problem. We also showed that the algorithm's ability of escaping from a local optima, that has a large BLOCKONES, is very important. Because it dominates the time complexity of finding a maximum clique. This paper further investigates this metric, and we will focus on analyzing the benefits of hybridizing the dynamic mutation approach with different local searches.

Based on the metric BLOCKONES, our main results are:

1) Hybridizing the dynamic mutation with the First-Improvement Local Search (FILS) enhances the algorithm's ability to escape from any local optimal solution x with $\texttt{BLOCKONES}(x) \geq 2$. Furthermore, the magnitude of this enhancement is super-polynomial if $\texttt{BLOCKONES}(x) = \omega(1)$, or even exponential if $\texttt{BLOCKONES}(x) = \Theta(n)$.
2) Hybridizing the dynamic mutation with the Best-Improvement Local Search (BILS) enhances the algorithm's ability to escape from any local optimal solution x with $\texttt{BLOCKONES}(x) = \Omega(\log\log n)$. Furthermore, the magnitude of this enhancement is super-polynomial if $\texttt{BLOCKONES}(x) = \Omega(\log n)$, or even exponential if $\texttt{BLOCKONES}(x) = \Theta(n)$.

The paper is structured as follows. In Section II, we state the (1+1) EA, and the (1+1) Dynamic Memetic Algorithm (DMA) with two different local searches—FILS and BILS. In Section III, we first define the metric BLOCKONES, then

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

analyze the upper bounds of each algorithm's ability to escape from a local optima and find a better solution. In Section IV, we show that hybridizing the dynamic mutation approach with the FILS enhances the algorithm's ability to escape from a local optima. In Section V, we show that hybridizing the dynamic mutation approach with the BILS enhances the algorithm's ability to escape from a local optima. 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) $f(n) = O(g(n)) \leftrightarrow \exists k > 0, \exists n_0, \forall n > n_0, f(n) \leq g(n) \cdot k$.
4) $f(n) = \Theta(g(n)) \leftrightarrow \exists k_1 > 0, \exists k_2 > 0, \exists n_0, \forall n > n_0, g(n) \cdot k_1 \leq f(n) \leq g(n) \cdot k_2$.
5) $\lim_{n\to\infty} (1 + 1/n)^n = e$.

A. The 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 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_{OL} for the Clique Problem.

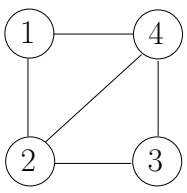
For a given graph $G = (V = \{v_1, v_2, ..., v_n\}, E)$, a bit string $x = (x_1, x_2, ..., x_n) \in \{0, 1\}^n$ defines a clique potential solution (an induced subgraph) where $x_i = 1$ represents that vertex v_i is selected. We say x represents a clique if each selected vertex in x is connected to all 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_{OL} (`ONEMAX` and `LackEdges`) is defined as follows:

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

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, $f_{\text{OL}}(1101) = 3$ because $x = (1101)$ is a clique consists of vertices 1, 2 and 4. $f_{\text{OL}}(1111) = -1$ 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 $f_{\text{OL}}(x)$.

B. Algorithms to be analyzed

The algorithms we will analyze on the Clique Problem are the (1+1) DMA with two local searches, FILS and BILS, respectively, and the (1+1) EA [5] (a version that does not include the dynamic mutation approach nor the local search). Note these algorithms all try to maximize the function $f = f_{\text{OL}}$. The time complexity analysis in this paper only looks at the number of evaluations of this fitness function f_{OL}. The algorithms are stated as below:

Algorithm 3. (1+1) EA for functions $f : \{0, 1\}^n \to \mathcal{R}$:

1) Initialize the mutation probability $p = 1/n$.
2) Choose $x \in \{0, 1\}^n$ uniformly at random.
3) $y := x$. Flip every bit in y with probability p.
4) If $f(y) \geqslant f(x)$ then $x := y$.
5) Stop if any stopping criterion is met; otherwise, go to to step 3.

The (1+1) Dynamic Memetic Algorithm (DMA) dynamically adjust the mutation probability in every generation. A template of the generic algorithm is as follows:

Algorithm 4. (1+1) DMA for functions $f : \{0, 1\}^n \to \mathcal{R}$

1) Initialize the mutation probability $p := 1/n$.
2) Choose $x \in \{0, 1\}^n$ uniformly at random.
3) $y := x$. Flip every bit in y with probability p.
4) $z := \texttt{LocalSearch}(y)$.
5) $p := \texttt{Dynamic}(p)$.
6) If $f(z) \geqslant f(x)$ then $x := z$.
7) Stop if any stopping criterion is met; otherwise, go to step 3.

We do not yet specify any stopping criterion since we will analyze both algorithms' ability of escaping from all local optimal solutions until it finds a global optimal solution. For the (1+1) DMA, the function `Dynamic` will be stated below and the function `LocalSearch` will be stated in the next Section.

The mutation in the DMA is mainly used for making a jump in the search space when the DMA has been stagnated at a local optimal solution. Meanwhile, the mutation probability $p = 1/n$ means that the expected number of flipped bits is one. So the Dynamic function in step 5 is chosen as below:

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

Note that this dynamic mutation approach has also been studied for EAs in [8].

C. Two local searches

The local search in MAs can have many variations. We will analyze a First-improvement Local Search (FILS) and a Best-improvement Local Search (BILS) in this paper.

Algorithm 5. FILS for a given string $x =\in \{0, 1\}^n$:

1) Generate a random permutation `Per` of length n.
2) $i := 0$, `NoImproveCount` $:= 0$.

3) $y := \texttt{flip}(x, \texttt{Per}[i])$.
4) If $f(y) > f(x)$ then $x := y$, $\texttt{NoImproveCount} := 0$.
5) $\texttt{NoImproveCount} := \texttt{NoImproveCount} + 1$.
6) $i := (i + 1) \bmod n$.
7) Stop and return x if $\texttt{NoImproveCount} = n$. Otherwise, go to step 3.

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

Example 6. Suppose the string $x = (0, 0, 0, 0)$, and $\texttt{Per} = (3, 2, 1, 4)$. So the FILS will first check a possible flipping for the third bit in x to get $y = (0, 0, 1, 0)$. If $f(y) > f(x)$ then $x = y$. This check sequence follows Per in a cyclic fashion. That is, after checking the fourth bit in x, the FILS will restart checking the third bit in x. In this example, the checking sequence is $3, 2, 1, 4, 3, 2, \cdots$.

Also note that the FILS moves to the first neighbor that improves the fitness value, so we have:

1) If the bit string after the mutation represents a clique, the FILS will take at most n fitness evaluations to flip bits from zeros to ones to find a maximal clique. Then it will take another n fitness evaluations to find out that there is no better neighbors, hence, stop.
2) If the bit string after the mutation does not represent a clique, the FILS will take at most n fitness evaluations to check all n bits to find a clique. During this time, for each bit, if its value is one, and also flipping this bit from one to zero will improve the fitness value, then the FILS will execute the flip. Hence, the FILS will find a clique within n fitness evaluations.

Therefore, the FILS will stop on the fitness functions f_{OL} within $3n$ fitness evaluations.

Unlike the FILS that moves to its neighbor as soon as the fitness value improves, the BILS evaluates all n neighbors before moving to one neighbor.

Algorithm 7. BILS 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 BestNeighborSet.
4) Go to step 1.

Note that $\texttt{Hamming}(x, y)$ is the number of different bits between x and y. Also note that the BILS will evaluate all n neighbors and then move to one of the best neighbor, thus we have:

1) If the bit string after the mutation represents a clique, the BILS will keep flipping bits from zeros to ones until it finds a local optimal clique, which takes at most n^2 fitness evaluations.
2) If the bit string after the mutation does not represent a clique, the BILS will keep flipping bits from ones to zeros until it finds a clique, which takes at most n^2 fitness evaluations.

Therefore, the BILS will stop on the fitness functions f_{OL} within $2n^2$ fitness evaluations.

In the rest of this paper, we will use (1+1) DMA_FILS to denote the algorithm (1+1) DMA using FILS as the local search, and use (1+1) DMA_BILS to denote the algorithm (1+1) DMA using BILS as the local search.

III. New Metric on Stagnation Analysis

Now we analyze how these algorithms cope with stagnations when they have been trapped into a local optimal clique. We first restate the metric BLOCKONES to measure the difficulty of escaping from a local optimal clique. Then we show that this metric will tell us that each algorithm has found a maximum clique with overwhelming high probability, which can be applied as a stop criterion for engineering design.

Definition 8. 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. $f_{\text{OL}}(y) > f_{\text{OL}}(x)$ 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.

Theorem 9. *If the analyzed algorithms are stagnated at a local optimal solution x, where $t = \texttt{BLOCKONES}(x)$, the expected number of fitness evaluations to skip out of this local optimal solution and find a larger clique is bounded by*

1) $O\left(n^{2t+1}\right)$ *for the (1+1) EA.*
2) $O\left(n^{t+2} e^{2t} \log n\right)$ *for the (1+1) DMA_FILS.*
3) $O\left(\frac{n^{2t} e^{2t} \log n}{t^{2t+1}}\right)$ *for the (1+1) DMA_BILS.*

Proof. Since $t = \texttt{BLOCKONES}(x)$, there must exist a larger clique y such that $f_{\text{OL}}(y) = f_{\text{OL}}(x) + 1$ and $\sum_{i=1}^{n} \overline{y_i} x_i = t$. Now we look at the expected fitness evaluations that each algorithm will find this y. In order to find the y, let U_1 and U_2 be two sets of bits that each algorithm needs to flip from ones to zeros and flip from zeros to ones, respectively, i.e. $U_1 = \{i \mid x_i = 1 \text{ and } y_i = 0\}$ and $U_2 = \{j \mid x_j = 0 \text{ and } y_j = 1\}$. Furthermore, we know $|U_1| = t$ and $|U_2| = t + 1$.

Part 1. For the (1+1) EA.
The mutation needs to flip all bits in U_2 from zeros to ones, flip all bits in U_1 from ones to zeros, and keep all other bits unflipped. This probability is $p^{|U_1|+|U_2|}(1-p)^{n-|U_1|-|U_2|}$, which is $\Theta\left((1/n)^{2t+1}\right)$ when $p = 1/n$, $|U_1| = t$ and $|U_2| = t + 1$. Thus the upper bound is proved.

Part 2. For the (1+1) DMA_FILS.
Let the mutation flip all bits in U_2 from zeros to ones, and do not flip any other bits. The probability of this mutation occurring is $p^{t+1}(1-p)^{n-2t-1}$. Note that after this mutation, the new bit string does not represent a clique. Hence, the function $LackEdges$ in the fitness function will guide the FILS to flip some bits from ones to zeros in order to find a clique. Also note that any vertex, that is represented by a bit in U_1, is not adjacent to some vertices, that are represented by the bits in U_2. Therefore, after the mutation, flipping any bit in U_1 from one to zero will improve the fitness value. Furthermore, the sequence of the bits that will be flipped is determined by the sequence of the random permutation array. Therefore, if the FILS generates a permutation array, that will guide the FILS to check all bits in U_1 before checking any bit in U_2, then the FILS will flip all bits in U_1 from ones to zeros, hence, it finds y. The success probability to get this permutation array is $(1/t)^{t+1}$ (recall that $|U_1| = t$ and $|U_2| = t+1$).

Overall, the success probability of the (1+1) DMA_FILS escaping from x and finding a larger clique is at least $p^{t+1}(1-p)^{n-2t-1}(1/t)^{t+1}$. When $t/n \leq p < 2t/n$, we know that $p^{t+1} \geq (\frac{t}{n})^{t+1}$ and $(1-p)^{n-2t-1} > (1-\frac{2t}{n})^{n-2t-1} = \Omega\left(e^{-2t(n-2t-1)/n}\right)$. Therefore, the success probability of the (1+1) DMA_FILS escaping from x and finding a larger clique is $\Omega\left(n^{-t-1}e^{-2t}\right)$. Besides, the dynamic mutation approach will have one mutation probability p, with $t/n \leq p < 2t/n$, in every $\lceil \log n \rceil$ mutations; also, after each mutation, the FILS will take at most $3n$ fitness evaluations to find a local optimal solution (see Section II-C).

Part 3. For the (1+1) DMA_BILS.
Let the mutation flip t bits in U_2 from zeros to ones, flip $(t-2)$ bits in U_1 from ones to zeros, and do not flip any other bits (note that $|U_1| = t$ and $|U_2| = t+1$). The success probability of this mutation occurring is $\binom{t+1}{1}\binom{t}{2}p^{2t-2}(1-p)^{n-2t+2}$. When $t/n \leq p < 2t/n$, we know that $p^{2t-2} \geq (\frac{t}{n})^{2t-2}$ and $(1-p)^{n-2t+2} > (1-\frac{2t}{n})^{n-2t+2} = \Omega\left(e^{-2t(n-2t+2)/n}\right)$. Therefore, this probability is $\Omega\left(t^{2t+1}n^{-2t+2}e^{-2t}\right)$. Then, after this mutation, the bit string will have two bits in U_1 and t bits in U_2 to be ones, while all other bits will be zeros. Furthermore, we claim that the BILS will have a success probability of at least $1/4$ to flip the two bits in U_1 from ones to zeros (we will show this in the next paragraph). Hence, it will get a clique that has the same clique size as x, and that is a sub-clique of y. Therefore, the BILS will at least flip one more bit from zero to one, hence, it escapes from x and find a larger clique (this larger clique may not be y).

Overall, within $O\left(t^{-2t-1}n^{2t-2}e^{2t}\right)$ mutations, where the mutation probability p satisfies $t/n \leq p < 2t/n$, the (1+1) DMA_BILS is expected to escape from x and find a larger clique. Again, the dynamic mutation approach will have one mutation with $t/n \leq p < 2t/n$ in every $\lceil \log n \rceil$ mutations; also, after each mutation, the BILS will take at most $2n^2$ fitness evaluations to find a local optimal solution (see Section II-C). Therefore, the upper bound for the (1+1) DMA_BILS is proved.

Now we show that if the mutation has flipped t bits in U_2 from zeros to ones, $(t-2)$ bits in U_1 from ones to zeros, and keep all other bits un-flipped, then the BILS has a probability of at least $1/4$ that flips the other two bits in U_1 from ones to zeros to find a sub-clique of y. Note that to illustrate our proof easily, we treat each bit in the bit string as a vertex, for example, we will use "x_1 is not adjacent to x_2" to denote that the vertex that is represented by x_1 is not adjacent to the vertex that is represented by x_2.

Let x_i and x_j be the two bits in U_1 that are not flipped from ones to zeros by the mutation. Let U_2^t be the t bits in U_2 that have been flipped from zeros to ones by the mutation. Then, we know that x_i is adjacent to x_j because they are two vertices in the clique of x, and all bits in U_2^t are adjacent to each other because they are the vertices in the clique of y.

Furthermore, we claim that x_i and x_j are not adjacent to some vertices in U_2^t, respectively. Otherwise, if any vertex, suppose x_i, is adjacent to all vertices in U_2^t, then we can get a larger clique by flipping the bit x_j from one to zero, and this larger clique is obtained by flipping $(t-1)$ bits in x from ones to zeros, which is conflicted with $t = \texttt{BLOCKONES}(x)$.

Therefore, after the mutation, the BILS will find out that flipping the bit x_i or x_j from one to zero will improve the fitness value by reducing the value of the function `LackEdges`. Meanwhile, it will also find out that flipping some bits in U_2^t will also improve the fitness value. Then, the result may be:

1) If any vertex of x_i or x_j is not adjacent to three or more vertices in U_2^t, then the BILS will first flip the x_i or x_j from one to zero because that is the best neighbor of the current bit string. Note that any vertex in U_2^t is not adjacent to at most two vertices (x_i and x_j).
2) If there is only one vertex in $\{x_i, x_j\}$ that is not adjacent to two vertices in U_2^t, then there is at most one vertex in U_2^t that is not adjacent to both the vertices x_i and x_j. Therefore, the BILS will have at least $1/2$ chance that will first flip a bit from $\{x_i, x_j\}$ from one to zero.
3) If both the vertices of x_i and x_j are not adjacent to two vertices in U_2^t, then there is at most two vertices in U_2^t that are not adjacent to both the vertices x_i and x_j. Therefore, the BILS still has at least $1/2$ chance that will first flip a bit from $\{x_i, x_j\}$ from one to zero.
4) If both the vertices of x_i and x_j are not adjacent to only one vertex in U_2^t, respectively. First of all, we claim that, the vertex in U_2^t that is not adjacent to x_i, and the vertex in U_2^t that is not adjacent to x_j, cannot be the same vertex. Otherwise, we can obtain a clique by flipping that bit in U_2^t from one to zero, where this clique has a larger clique size than x and this clique only flips $(t-2)$ bits in x from ones to zeros, which is conflicted with $t = \texttt{BLOCKONES}(x)$. Therefore, since x_i and x_j are not adjacent to two different vertices in U_2^t, respectively, the BILS still has at least $1/2$ chance that will first flip a bit from $\{x_i, x_j\}$ from one to zero.

Suppose the BILS first flips a bit of x_i or x_j from one to zero, which has a chance greater than $1/2$ as shown above. Then, in the new bit string, the only bit in U_1, that is one, is not adjacent to at least one vertex in U_2^t (we have shown above that both x_i and x_j are not adjacent to some vertices in U_2^t). Thus, the BILS will have at least $1/2$ chance to flip

the other bit in U_1 from one to zero. Overall, the BILS will have at least $1/4$ probability to flip the two bits in U_1 from ones to zeros. □

Note to evaluate the function $\texttt{BLOCKONES}(x)$ requires one to compute all cliques in the graph, which is too expensive. However, we know that $\texttt{BLOCKONES}(x) \leq \texttt{ONEMAX}(x)$, and we will motivate the metric of $\texttt{BLOCKONES}(x)$ by the corollary below.

Corollary 10. *If the analyzed algorithms are stagnated at a local optimal solution x for the following number of fitness evaluations, then with overwhelming probability, this x is a maximum clique of the graph. let $t_m = \textit{ONEMAX}(x)$, we have:*

1) n^{2t_m+2} *for the (1+1) EA.*
2) $n^{t_m+3}e^{2t_m}\log n$ *for the (1+1) DMA_FILS.*
3) $\frac{n^{2t_m+1}e^{2t_m}\log n}{t_m^{2t_m+1}}$ *for the (1+1) DMA_BILS.*

Proof. If any of the above algorithms is expected to escape from x and find a larger clique in $O(g(n))$ fitness evaluations with overwhelming probability, then the probability that this event will happen within $n \cdot g(n)$ fitness evaluations is exponentially close to one. That is to say, if this algorithm is stagnated at a local optimal solution for more than $n \cdot g(n)$ fitness evaluations, then with overwhelming probability it will not find any larger clique in the future. Hence, with overwhelming probability x is a maximum clique, due to the fact that the algorithm is expected to find a maximum clique within $O(n^n)$ mutations with the mutation probability $p = 1/n$. Therefore, according to Definition 8, $\texttt{BLOCKONES}(x) \leq t_m$, and because of Theorem 9, the corollary is proved. □

IV. Benefit of hybridizing the dynamic mutation with the FILS

In our previous study [2], we showed that the ability of jumping out of a local optimal clique x with a large $\texttt{BLOCKONES}(x)$ is very important. This is because of that it is the most time consuming part and dominates the time complexity of finding a maximum clique. This finding brings us a new question—what is the benefit of hybridizing the dynamic mutation approach with a local search in terms of escaping a local optimal clique x with a large $\texttt{BLOCKONES}(x)$?

In this section, we will compare the (1+1) DMA_FILS with the (1+1) EA, in order to show the benefit of hybridizing the dynamic mutation with the FILS. We will show that this hybridization enhances the algorithm's ability to escape from any local optimal solution x with $\texttt{BLOCKONES}(x) \geq 2$. Furthermore, the magnitude of this enhancement is super-polynomial if $\texttt{BLOCKONES}(x) = \omega(1)$, or even exponential if $\texttt{BLOCKONES}(x) = \Theta(n)$.

Lemma 11. *Suppose both the (1+1) DMA_FILS and the (1+1) EA have been stagnated at a local optimal clique x, and there exists another clique y with $f_{\textit{OL}}(y) \geq f_{\textit{OL}}(x)$. Let U_1 be the set of all bits that need to be flipped from ones to zeros in order to find y, i.e. $U_1 = \{i \mid x_i = 1, y_i = 0$ and $1 \leq i \leq n\}$. Then, the ratio of a) the probability that the (1+1) DMA_FILS finds the bit string y by one mutation and the following FILS, and b) the probability that the (1+1) EA finds the bit string y by one mutation, is $\Omega\left(n^{|U_1|}\right)$.*

Proof. Let U_2 be the set of all bits that need to be flipped from zeros to ones in order to find y, i.e. $U_2 = \{i \mid x_i = 0, y_i = 1$ and $1 \leq i \leq n\}$, then we have $|U_2| \geq |U_1|$. Hence, to jump from x to y, we need to flip all bits in U_1 from ones to zeros, and flip all bits in U_2 from zeros to ones. Now we define $\texttt{Prob}^{\texttt{EA}}_{(x\to y)}$ and $\texttt{Prob}^{\texttt{DMA\_FILS}}_{(x\to y)}$ as below:

Part 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:

$$\texttt{Prob}^{\texttt{EA}}_{(x\to y)} = p^{|U_1|}p^{|U_2|}(1-p)^{n-|U_1|-|U_2|}.$$

This probability is $\Theta\left(\frac{1}{n^{|U_1|+|U_2|}}\right)$ when $p = 1/n$.

Part 2. Let $\texttt{Prob}^{\texttt{DMA\_FILS}}_{(x\to y)}$ be the probability of the (1+1) DMA_FILS jumping from x to y using one mutation and the following FILS. We measure the way that satisfies the following two conditions:

1) The mutation will flip all bits in U_2 from zeros to ones and keep all other bits un-flipped. The success probability of this mutation occurring is $p^{|U_2|}(1-p)^{n-|U_2|}$.
2) The permutation array will guide the FILS to check all bits in U_1 before checking any bit in U_2. The success probability of the random permutation array satisfying this condition is $\left(\frac{1}{|U_1|}\right)^{|U_2|}$.

Therefore, in this way, the mutation will create a bit string that does not represent a clique, then the FILS will flip all bits in U_1 from ones to zeros, hence, find the bit string y. And the success probability is $p^{|U_2|}(1-p)^{n-|U_2|}(\frac{1}{|U_1|})^{|U_2|}$. Recall that the dynamic mutation approach will double the mutation probability until it reaches $1/2$, therefore, in every $\lceil \log n \rceil$ mutations, we will have one mutation that has the mutation probability p' with $\frac{|U_1|}{n} \leq p' < \frac{2|U_1|}{n}$. Hence, we have:

$$\texttt{Prob}^{\texttt{DMA\_FILS}}_{(x\to y)} > p'^{|U_2|}(1-p')^{n-|U_2|}\left(\frac{1}{|U_1|}\right)^{|U_2|},$$

where $p'^{|U_2|} \geq (|U_1|/n)^{|U_2|}$ and $(1-p')^{n-|U_2|} > (1-2|U_1|/n)^{n-|U_2|}$. Thus, this probability $\Omega\left(\frac{1}{n^{|U_2|}}\frac{1}{c^{|U_1|}}\right)$ for a constant $c \geq 1$.

Overall, we have:

$$\frac{\texttt{Prob}^{\texttt{DMA\_FILS}}_{(x\to y)}}{\texttt{Prob}^{\texttt{EA}}_{(x\to y)}} = \Omega\left(\left(\frac{n}{c}\right)^{|U_1|}\right).$$ □

Note that Lemma 11 restricts that both the (1+1) EA and the (1+1) DMA_FILS need to directly jump from x to y without visiting any other intermediate cliques. However, the algorithms may jump to some other intermediate cliques first, and eventually move to y. Now we consider both situations (directly jumping and via some intermediate cliques) and show that the (1+1) DMA_FILS is expected to escape from x and find a larger clique faster than the (1+1) EA.

Theorem 12. *Suppose both the (1+1) DMA_FILS and the (1+1) EA have been stagnated at a local optimal clique x*

with $t = \texttt{BLOCKONES}(x)$. Then the ratio of a) the probability that the (1+1) DMA_FILS escapes from x and finds a larger clique, and b) the probability that the (1+1) EA escapes from x and finds a larger clique, is $\Omega\left(n^t/c^t\right)$ for a constant c.

Proof. To prove the theorem, we show that if both algorithms have been stagnated at x, then for any clique y such that y is the first clique that the (1+1) EA will find with $f_{\texttt{OL}}(y) > f_{\texttt{OL}}(x)$ (escape from x), the probability of the (1+1) DMA_FILS finding the y is larger than the probability of the (1+1) EA finding the y. Furthermore, the ratio of these probabilities is $\Omega\left(n^t/c^t\right)$.

In order to find the y, let U_1 and U_2 be two sets of bits that need to be flipped from ones to zeros and from zeros to ones respectively, i.e. $U_1 = \{i \mid x_i = 1, y_i = 0 \text{ and } 1 \leq i \leq n\}$ and $U_2 = \{i \mid x_i = 0, y_i = 1 \text{ and } 1 \leq i \leq n\}$. Then we know that $|U_2| > |U_1| \geq t$.

Since the (1+1) EA only accepts a new solution if its fitness is greater than or equal to the current solution, it can only escape from the clique x to the 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. Note y is the first solution that both algorithms have found, and that is better than x.

Proof of case (a), by directly mutating from x to y.

According to Lemma 11, and because $|U_1| \geq t$, this ratio is $\Omega\left(n^t/c^t\right)$ for a constant $c \geq 1$.

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 λ-length path $x^1 \to x^2 \to \cdots \to x^{\lambda-1} \to x^\lambda$, where $x^1 = x$, $x^\lambda = y$, $\lambda \geq 2$ and each x^i with $1 \leq i \leq \lambda - 1$ is a bit string with $f_{\texttt{OL}}(x) = f_{\texttt{OL}}(x^i)$. We prove this part by comparing the probability of the (1+1) DMA_FILS jumping along this $\texttt{Path}_\lambda$ to reach y and the probability of the (1+1) EA jumping along this $\texttt{Path}_\lambda$ to reach y, and showing that the ratio of these two probabilities supports our theorem.

For any jump from x^i to x^{i+1} $(1 \leq i \leq \lambda - 1)$ in $\texttt{Path}_\lambda$, let x^i_j denote the j-th bit of the bit string x^i, and let T_i be the set of bits that need to be flipped from ones to zeros in order to find x^{i+1}, i.e. $T_i = \{j \mid x^i_j = 1, x^{i+1}_j = 0 \text{ and } 1 \leq j \leq n\}$. So according to Lemma 11, the ratio of a) the probability that the (1+1) DMA_FILS finds x^{i+1} from x^i, and b) the probability that the (1+1) EA finds x^{i+1} from x^i is $\Omega\left(n^{|T_i|}/c^{|T_i|}\right)$ for a constant $c \geq 1$.

Overall, the ratio of a) the probability that the (1+1) DMA_FILS jumps along $\texttt{Path}_\lambda$ to find the y, and b) the probability that the (1+1) EA jumps along $\texttt{Path}_\lambda$ to find the y is:

$$\Omega\left(\prod_{i=1}^{\lambda-1}\left(\frac{n}{c}\right)^{|T_i|}\right).$$

Recall that U_1 is the set of bits that need to be flipped from ones to zeros in order to find the y from the x, thus $\sum_{i=1}^{\lambda-1}|T_i| \geq |U_1| \geq t$, thus the overall ratio is greater than $\Omega(n^t/c^t)$. □

Corollary 13. *Suppose both the (1+1) DMA_FILS and the (1+1) EA have been stagnated at a local optimal clique x with $t = \texttt{BLOCKONES}(x)$. Let $E_{\texttt{EA}}(x)$ be the expected fitness evaluations of the (1+1) EA escaping from x and find a larger clique, and let $E_{\texttt{DMA\_FILS}}(x)$ be the expected fitness evaluations of the (1+1) DMA_FILS escaping from x and find a larger clique. Then $E_{\texttt{EA}}(x)$ is larger than $E_{\texttt{DMA\_FILS}}(x)$ when $t \geq 2$. Furthermore, the ratio of the $E_{\texttt{EA}}(x)$ and the $E_{\texttt{DMA\_FILS}}(x)$ is:*

1) *Exponentially large if $t = \Theta(n)$.*
2) *Super-polynomially large if $t = \omega(1)$.*
3) *Polynomially large if $t = \Theta(1)$ and $t \geq 2$.*

Proof. According to Theorem 12, the probability of the (1+1) DMA_FILS escaping from x and finding a larger clique is $\Omega\left(n^t/c^t\right)$ times the probability of the (1+1) EA escaping from x and finding a larger clique. However, note that the dynamic mutation approach will take $\lceil \log n \rceil$ mutations to iterate the mutation probability between $1/n$ and $1/2$, and each mutation is followed by one FILS that will take at most $3n$ fitness evaluations. Therefore, the overall ratio is $\Omega\left(\frac{n^t}{c^t n \log n}\right)$, which is greater than one when $t \geq 2$, super-polynomial when $t = \omega(1)$ and exponential when $t = \Theta(1)$. □

V. Benefit of hybridizing the dynamic mutation with the BILS

In this section, we will compare the (1+1) DMA_BILS with the (1+1) EA, in order to show the benefit of hybridizing the dynamic mutation with the BILS. We will show that this hybridization enhances the algorithm's ability to escape from any local optimal solution x with $\texttt{BLOCKONES}(x) = \Omega(\log \log n)$. Furthermore, the magnitude of this enhancement is super-polynomial if $\texttt{BLOCKONES}(x) = \Omega(\log n)$, or even exponential if $\texttt{BLOCKONES}(x) = \Theta(n)$.

Lemma 14. *Suppose both the (1+1) DMA_BILS and the (1+1) EA have been stagnated at a local optimal clique x, and there exists another clique y with $f_{\texttt{OL}}(y) \geq f_{\texttt{OL}}(x)$. Let U_1 be the set of all bits that need to be flipped from ones to zeros in order to find y, i.e. $U_1 = \{i \mid x_i = 1, y_i = 0 \text{ and } 1 \leq i \leq n\}$. Then, the ratio of a) the probability that the (1+1) DMA_BILS finds the bit string y by one mutation and the following BILS, and b) the probability that the (1+1) EA finds the bit string y by one mutation, is $\Omega\left(n^2|U_1|^{|U_1|}\right)$.*

Proof. Let U_2 be the set of all bits that need to be flipped from zeros to ones in order to find y, i.e. $U_2 = \{i \mid x_i = 0, y_i = 1 \text{ and } 1 \leq i \leq n\}$, then we have $|U_2| \geq |U_1|$. Hence, to jump from x to y, we need to flip all bits in U_1 from ones to zeros, and flip all bits in U_2 from zeros to ones. Now we define $\texttt{Prob}^{\texttt{EA}}_{(x \to y)}$ and $\texttt{Prob}^{\texttt{DMA\_BILS}}_{(x \to y)}$ as below:

Part 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:

$$\texttt{Prob}^{\texttt{EA}}_{(x \to y)} = p^{|U_1|}p^{|U_2|}(1-p)^{n-|U_1|-|U_2|}.$$

This probability is $\Theta\left(\frac{1}{n^{|U_1|+|U_2|}}\right)$ when $p = 1/n$.

Part 2. Let $\texttt{Prob}^{\texttt{DMA\_BILS}}_{(x\to y)}$ be the probability of the (1+1) DMA_BILS jumping from x to y using one mutation and the following BILS. We measure the way that satisfies the following two conditions:

1) The mutation will flip all bits in U_2 from zeros to ones, flip $(|U_1|-2)$ bits in U_1 from ones to zeros, and keep all other bits un-flipped. The success probability of this mutation occurring is $\binom{|U_1|}{2}p^{|U_1|+|U_2|-2}(1-p)^{n-|U_1|-|U_2|+2}$.
2) The BILS will find out that the bit string after the mutation does not represent a clique, and flip the two bits in U_1 from ones to zeros, hence, it finds the y. The success probability of achieving this is at least $1/4$ (see the proof of Part 3 in Theorem 9).

Therefore, the success probability is $\binom{|U_1|}{2}p^{|U_1|+|U_2|-2}(1-p)^{n-|U_1|-|U_2|+2}$. Recall that the dynamic mutation approach will double the mutation probability until it reaches $1/2$. Hence, in every $\lceil \log n \rceil$ mutations, we will have one mutation that has the mutation probability p' with $\frac{|U_1|}{n} \leq p' < \frac{2|U_1|}{n}$. So we have:

$$\texttt{Prob}^{\texttt{DMA\_BILS}}_{(x\to y)} > \binom{|U_1|}{2}p'^{|U_1|+|U_2|-2}(1-p')^{n-|U_1|-|U_2|+2},$$

which is $\Omega\left(\frac{|U_1|^{|U_1|+|U_2|}}{n^{|U_1|+|U_2|-2}}\frac{1}{c^{|U_1|}}\right)$ for a constant $c>1$.

Overall, we have:

$$\frac{\texttt{Prob}^{\texttt{DMA\_BILS}}_{(x\to y)}}{\texttt{Prob}^{\texttt{EA}}_{(x\to y)}} = \Omega\left(n^2|U_1|^{|U_1|+|U_2|}c^{-|U_1|}\right) = \Omega\left(n^2|U_1|^{|U_1|}\right).$$

□

Note that Lemma 14 restricts that both the (1+1) EA and the (1+1) DMA_BILS need to directly jump from x to y without visiting any other intermediate cliques. However, the algorithms may jump to some other intermediate cliques first, and eventually move to y. Now we consider both situations and show that the (1+1) DMA_BILS is expected to escape from x and find a larger clique faster than the (1+1) EA when $\texttt{BLOCKONES}(x) = \omega(1)$.

Theorem 15. *Suppose both the (1+1) DMA_BILS and the (1+1) EA have been stagnated at a local optimal clique x with $t = \texttt{BLOCKONES}(x)$. Then, the ratio of a) the probability that the (1+1) DMA_BILS escapes from x and finds a larger clique, and b) the probability that the (1+1) EA escapes from x and finds a larger clique, is super-polynomial if $t = \Omega(\log n)$, or is exponential if $t = \Theta(n)$.*

Proof. To prove the theorem, we show that if both algorithms have been stagnated at x, then for any clique y such that y is the first clique that the (1+1) EA will find with $f_{\texttt{OL}}(y) > f_{\texttt{OL}}(x)$ (escape from x), the probability of the (1+1) DMA_BILS finding the y is larger than the probability of the (1+1) EA finding the y. Furthermore, the ratio of these probabilities is super-polynomial if $t = \Omega(\log n)$.

In order to find the y, let U_1 and U_2 be two sets of bits that need to be flipped from ones to zeros and from zeros to ones, respectively, i.e. $U_1 = \{i \mid x_i = 1, y_i = 0 \text{ and } 1 \leq i \leq n\}$ and $U_2 = \{i \mid x_i = 0, y_i = 1 \text{ and } 1 \leq i \leq n\}$. Then we know that $|U_2| > |U_1| \geq t$.

Since the (1+1) EA only accepts a new solution if its fitness is greater than or equal to the current solution, it can only escape from the clique x to the 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. Note y is the first solution that both algorithms have found, and that is better than x.

Proof of case (a), by directly mutating from x to y.

According to Lemma 14, and because $|U_1| \geq t$, this ratio is $\Omega\left(n^2t^t\right)$. Clearly this ratio is exponential if $t = \Theta(n)$. Furthermore, we claim that it is super-polynomial if $t = \Omega(\log n)$. This is because that when n approaches infinite, there exists a constant c, such that $t > \log(n/c)$. Then we have:

$$t^t = \left(2^{\log t}\right)^t = 2^{t\cdot\log t} > 2^{\log(n/c)\cdot\log\log(n/c)}$$
$$= 2^{\log\left((n/c)^{\log\log(n/c)}\right)} = (n/c)^{\log\log(n/c)},$$

which is super-polynomial.

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 λ-length path $x^1 \to x^2 \to \cdots \to x^{\lambda-1} \to x^\lambda$, where $x^1 = x$, $x^\lambda = y$, $\lambda \geq 2$ and each x^i with $1 \leq i \leq \lambda - 1$ is a bit string with $f_{\texttt{OL}}(x) = f_{\texttt{OL}}(x^i)$. We prove this part by comparing the probability of the (1+1) DMA_BILS jumping along this $\texttt{Path}_\lambda$ to reach y and the probability of the (1+1) EA jumping along this $\texttt{Path}_\lambda$ to reach y, and showing that the ratio of these two probabilities supports our theorem.

For any jump from x^i to x^{i+1} $(1 \leq i \leq \lambda - 1)$ in $\texttt{Path}_\lambda$, let x^i_j denote the j-th bit of the bit string x^i, and let T_i be the set of bits that need to be flipped from ones to zeros in order to find x^{i+1}, i.e. $T_i = \{j \mid x^i_j = 1, x^{i+1}_j = 0 \text{ and } 1 \leq j \leq n\}$. So according to Lemma 14, the ratio of a) the probability that the (1+1) DMA_BILS finds x^{i+1} from x^i, and b) the probability that the (1+1) EA finds x^{i+1} from x^i is $\Omega\left(n^2|T_i|^{|T_i|}\right)$.

Overall, the ratio of a) the probability that the (1+1) DMA_FILS jumps along $\texttt{Path}_\lambda$ to find the y, and b) the probability that the (1+1) EA jumps along $\texttt{Path}_\lambda$ to find the y is:

$$\Omega\left(n^{2\lambda-2}\prod_{i=1}^{\lambda-1}|T_i|^{|T_i|}\right).$$

Note that if $\lambda = \omega(1)$, this ratio is super-polynomial. Therefore, we only show that, in the case of $\lambda = \Theta(1)$, this ratio is super-polynomial if $t = \Omega(\log n)$, or even exponential if $t = \Theta(n)$. Recall that U_1 is the set of bits that need to be flipped from ones to zeros in order to find the y from the x, thus $\sum_{i=1}^{\lambda-1}|T_i| \geq |U_1| \geq t$. Therefore, since we only consider the case of $\lambda = \Theta(1)$, there must exist at least one T_i with $T_i = \Theta(t)$. Thus, $|T_i|^{|T_i|}$ is exponential if $t = \Theta(n)$, and is super-polynomial if $t = \Omega(\log n)$ (in case (a), we showed t^t is super-polynomial if $t = \Omega(\log n)$). □

Corollary 16. *Suppose both the (1+1) DMA_BILS and the (1+1) EA have been stagnated at a local optimal clique x*

with $t = \texttt{BLOCKONES}(x)$. Let $E_{\texttt{EA}}(x)$ be the expected fitness evaluations of the (1+1) EA escaping from x and find a larger clique, and let $E_{\texttt{DMA\_BILS}}(x)$ be the expected fitness evaluations of the (1+1) DMA_BILS escaping from x and find a larger clique. Then $E_{\texttt{EA}}(x)$ is larger than $E_{\texttt{DMA\_BILS}}(x)$ when $t = \Omega(\log\log n)$. Furthermore, the ratio of the $E_{\texttt{EA}}(x)$ and the $E_{\texttt{DMA\_BILS}}(x)$ is super-polynomially large if $t = \Omega(\log n)$.

Proof. Note that the BILS will take at most $2n^2$ fitness evaluations to find a local optimal clique, and the dynamic mutation approach will take $\lceil \log n \rceil$ mutations to iterate the mutation probability from $1/n$ to $1/2$. Therefore, the benefit of the (1+1) DMA_BILS will be downgraded by a factor of $n^2 \log n$.

When $t = \Omega(\log n)$, the ratio of the probabilities in Theorem 15 is super-polynomial, which is still super-polynomial after dividing by a factor of $n^2 \log n$.

When $t = \Omega(\log\log n)$, to show that $E_{\texttt{EA}}(x)$ will be larger than $E_{\texttt{DMA\_BILS}}(x)$, we need to prove that the ratio of the probabilities in Theorem 15 is $\omega\left(n^2 \log n\right)$. Recall the proof of both cases (a) and (b) for Theorem 15, the ratios of the probabilities are both $\Omega\left(n^2 t^t\right)$, so we only need to show that $t^t = \omega(\log n)$ when $t = \Omega(\log\log n)$. Since $t = \Omega(\log\log n)$, when n approaches infinite, there exists a constant c such that $t > \log\log(n/c)$. Then we have:

$$\begin{aligned} t^t &= \left(2^{\log t}\right)^t = 2^{t\cdot\log t} > 2^{\log\log(n/c)\cdot\log\log\log(n/c)} \\ &= 2^{\log\left((\log(n/c))^{\log\log\log(n/c)}\right)} = (\log(n/c))^{\log\log\log(n/c)} \\ &= \omega(\log n). \end{aligned}$$

□

VI. Conclusions and Future Work

This paper is a further investigation based on our previous study in [2]. We focus on analyzing the benefits of hybridizing the dynamic mutation approach with two different local searches, best-improvement and first-improvement, respectively. We showed that this hybridization enhances the Memetic Algorithm's ability to escape from a local optima and find a better solution. In detail, based on the metric BLOCKONES (minimum number of ones that are blocking each algorithm to find a larger clique), for any local optimal clique x with $t = \texttt{BLOCKONES}(x)$, we showed that:

1) Hybridizing the dynamic mutation with the FILS enhances the algorithm's ability to escape from x if $t \geq 2$. Furthermore, the magnitude of this enhancement is super-polynomial if $t = \omega(1)$, or even exponential if $t = \Theta(n)$.
2) Hybridizing the dynamic mutation with the BILS enhances the algorithm's ability to escape from x if $t = \Omega(\log\log n)$. Furthermore, the magnitude of this enhancement is super-polynomial if $t = \Omega(\log n)$, or even exponential if $t = \Theta(n)$.

A next step to investigate in the future is to find tight upper bounds or the lower bounds for the (1+1) DMA_FILS and the (1+1) DMA_BILS to escape from a local optima. Hopefully, that study will tell us if the the (1+1) DMA_FILS is more suitable for escaping from local optimal solutions.

References

[1] E. K. Burke and D. J. Landa-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. Springer, 2004, vol. 166, pp. 289–312.

[2] M. J. Dinneen and K. Wei, "A (1+1) adaptive memetic algorithm for the maximum clique problem," in *Proceedings of Congress on Evolutionary Computation*, CEC'13, vol. 1. IEEE, June 2013, pp. 1626–1634.

[3] ——, "On the analysis of a (1+1) adaptive memetic algorithm," in *Proceedings of Memetic Computing, MC2013*. IEEE, 2013, pp. 24–31.

[4] B. Doerr and M. Kunnemann, "How the (1+λ) evolutionary algorithm optimizes linear functions," in *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*, GECCO'13. ACM, 2013, pp. 1589–1596.

[5] S. Droste, T. Jansen, and I. Wegener, "On the analysis of the (1+1) evolutionary algorithm," *Theoretical Computer Science*, vol. 276, pp. 51–81, 2002.

[6] T. Jansen, K. A. De Jong, and I. Wegener, "On the choice of the offspring population size in evolutionary algorithms," *Evolution Computation*, vol. 13, no. 4, pp. 413–440, 2005.

[7] T. Jansen and I. Wegener, "On the analysis of evolutionary algorithms—A proof that crossover really can help," *Algorithmica*, vol. 34, no. 1, pp. 47–66, 2002.

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

[9] T. Kötzing, 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, 2011, pp. 989–996.

[10] O. Kramer, "Evolutionary self-adaptation: a survey of operators and strategy parameters," *Evolutionary Intelligence*, vol. 3, pp. 51–65, August 2010.

[11] F. Neri, C. Cotta, and P. Moscato, *Handbook of Memetic Algorithms*. Studies in Computational Intelligence, 2012, vol. 379.

[12] A. Q. Nguyen, A. M. Sutton, and F. Neumann, "Population size matters: Rigorous runtime results for maximizing the hypervolume indicator," in *Proceeding of the Fifteenth Annual Conference on Genetic and Evolutionary Computation Conference*, GECCO'13. ACM, 2013, pp. 1613–1620.

[13] C. Qian, Y. Yu, and Z.-H. Zhou, "An analysis on recombination in multi-objective evolutionary optimization," *Artificial Intelligence*, vol. 204, no. 0, pp. 99–119, 2013.

[14] D. Sudholt, "Local search in evolutionary algorithms: The impact of the local search frequency," *Algorithms and Computation*, vol. 4288, pp. 359–368, 2006.

[15] ——, "On the analysis of the (1+1) memetic algorithm," in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, GECCO'06. ACM, 2006, pp. 493–500.

[16] ——, "The impact of parametrization in memetic evolutionary algorithms," *Theoretical Computer Science*, vol. 410, no. 26, pp. 2511–2528, 2009.

[17] 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)*, LNCS, vol. 6506. Springer, 2010, pp. 340–352.

[18] K. Wei and M. J. Dinneen, "Runtime analysis comparison of two fitness functions on a memetic algorithm for the clique problem," in *Proceedings of Congress on Evolutionary Computation*, CEC'14. IEEE, July 2014, World Congress on Computaitonal Intelligence. To appear.

[19] ——, "Runtime analysis to compare best-improvement and first-improvement in memetic algorithms," in *Proceeding of the 16th Annual Conference on Genetic and Evolutionary Computation Conference*, GECCO'14. ACM, July 2014, To appear.

[20] C. Witt, "Runtime analysis of the (μ + 1) EA on simple pseudo-boolean functions," *Evolutionary Computation*, vol. 14, no. 1, pp. 65–86, 2006.