SCIENCE CHINA
Information Sciences Manuscript

• **RESEARCH PAPER** •

January 2015, Vol. 58 xxxxxx:1–xxxxxx:17
doi: xxxxxxxxxxxxxx

Variable solution structure can be helpful in evolutionary optimization

QIAN Chao, YU Yang* & ZHOU Zhi-Hua

National Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023*, China;*
Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 210023*, China*

Received January 1, 2015; accepted January 1, 2015; published online January 1, 2015

Abstract Evolutionary algorithms are a family of powerful heuristic optimization algorithms where various representations have been used for solutions. Previous empirical studies have shown that for achieving a better efficiency of evolutionary optimization, it is often helpful to adopt rich representations (e.g., trees and graphs) rather than ordinary representations (e.g., binary coding). Such a recognition, however, has little theoretical justifications. In this paper, we present a running time analysis on genetic programming. In contrast to previous theoretical efforts focused on simple synthetic problems, we study two classical combinatorial problems, the *maximum matching* and the *minimum spanning tree* problems. Our theoretical analysis shows that evolving tree-structured solutions is much more efficient than evolving binary vector encoded solutions, which is also verified by experiments. The analysis discloses that variable solution structure might be helpful in evolutionary optimization when the solution complexity can be well controlled.

Keywords evolutionary algorithms, genetic programming, maximum matching, minimum spanning tree, running time, computational complexity

Citation Qian C, Yu Y, Zhou Z-H. Variable solution structure can be helpful in evolutionary optimization. Sci China Inf Sci, 2015, 58: xxxxxx(17), doi: xxxxxxxxxxxxxx

1 Introduction

Evolutionary algorithms (EAs) [1] are a family of heuristic optimization algorithms that have been applied to various industrial tasks. In general, EAs follow the procedure of reproduction and selection cycles to evolve solutions, where a solution is represented using certain data structure and operators are applied to reproduce new solutions. For example, genetic algorithms (GAs) [2] usually represent solutions by binary vectors and employ bit-wise mutation and crossover operators to reproduce new solutions; genetic programming algorithms (GPs) [3] usually represent solutions using tree structures, and employ node-wise mutation and crossover operators accordingly.

Many previous empirical studies have shown the importance of solution representation in evolutionary optimization [4, 5]. For problems where solutions have complex structures, such as in evolving computer programs, electronic design, and symbolic regression, it is intuitive that structured representations of solutions are more natural and efficient than the vector representation. This intuition is supported by

* Corresponding author (email: yuy@lamda.nju.edu.cn)

empirical studies which have shown that GPs, ant colony optimization (ACO) algorithms, and particle swarm optimization (PSO) algorithms with variable-structured representation are able to produce better optimization performances [6–9]. However, such intuition has little theoretical justification [10]. This paper aims at analyzing the effect of solution representations through studying GPs on two representative combinatorial problems with structured solutions, the *maximum matching (MM)* and the *minimum spanning tree (MST)* problems.

1.1 Related work

As a kind of optimization algorithms, the foremost theoretical aspect is the running time complexity, which characterizes how soon an algorithm can solve a problem. Due to their complicated behaviors, the running time analysis of GPs started just 4 years ago. A simple GP, (1+1)-GP which employs population size 1, has been studied on some simple synthetic problems, including two separable model problems *Order* and *Majority* [11], the *Sorting* problem [12], the *Max* problem [13], and the weighted *Order* problem [14]. Kötzing et al. [15] also analyzed GP under the probably approximately correct (PAC) learning framework. Recently, SMO-GP (a simple multi-objective GP algorithm) has been analyzed on some two-objective optimization problems, which are transformed from the analyzed single-objective problems *Order*, *Majority*, and *Sorting* [14, 16, 17]. Despite the fast progress in building the theoretical foundation of GPs, these results are built on simple synthetic problems rather than real combinatorial problems.

For the running time analysis of GAs, many results were first reported on simple synthetic problems [18–20]. Later, the analysis emerged on classic combinatorial optimization problems, for example, MM and MST [21]. Compared with synthetic problems, the analysis on combinatorial problems can be viewed as an important progress, since it goes a step toward analyzing EAs on real problems.

For the MM problem, Giel and Wegener [22] first proved that (1+1)-EA (a simple GA with population size 1) finds a $(1+\epsilon)$-approximation of a MM in $O(m^{2\lceil \epsilon^{-1}\rceil})$ expected running time (ERT), where m is the number of edges of the given graph. This shows that GA is a polynomial-time randomized approximation scheme for this problem. Some running time bounds on concrete graphs have also been derived, for example, the polynomial ERT on path and tree graphs and the exponential running time on a class of bipartite graphs [22, 23].

For the MST problem, Neumann and Wegener [24] first proved that (1+1)-EA needs $O(m^2(\log n + \log w_{\max}))$ ERT, where m, n, and $w_{\max}$ are the number of edges, the number of nodes, and the maximum edge weight, respectively. Later, Doerr et al. [25] gave a coefficient for this bound, that is, $2em^2(1+\ln m + \ln w_{\max})$, using multiplicative drift analysis. Raidl et al. [26] also showed that using a biased mutation operator favoring edges with a small weight can drastically speed up finding a MST on complete graphs. By formulating the MST problem as a two-objective optimization problem, Neumann and Wegener [27] proved that GSEMO (a simple multi-objective GA) solves it in $O(mn(n + \log w_{\max}))$ expected time, which is better than (1+1)-EA for dense graphs (e.g., $m \in \Theta(n^2)$).

1.2 Our contribution

In this paper, we present a running time analysis of GPs. On the shoulder of previous theoretical studies analyzing GPs on simple synthetic problems (e.g., [11,12]), we study the MM and the MST problems; this is the first time the running time of GPs has been analyzed on combinatorial problems. We prove that the running time bound of (1+1)-GP on MM is $O((\frac{3m\min\{m,n\}}{2})^{\lceil \epsilon^{-1}\rceil})$ for $(1+\epsilon)$-approximation, and that on MST is $O(mn(\log n + \log w_{\max}))$ for exact solutions; these are better than the corresponding best running time bounds of (1+1)-EA proved so far as listed in Table 1, because the number of nodes n is usually smaller than the number of edges m. We also prove that the running time bound of SMO-GP on MST is $O(mn^2)$, which is better than the running time bound $O(mn(n + \log w_{\max}))$ of GSEMO reported in literature [27]. Our results provide theoretical evidence to support the usefulness of rich representations in evolutionary optimization. These findings are also verified by experiments in this paper. From the analysis, we find that the variable solution structure might be helpful in evolutionary optimization if the solution complexity can be well controlled.

Table 1 Comparison of ERT between GAs and GPs on the MM and the MST problems.

	MM	MST
GA: (1+1)-EA	$O(m^{2\lceil\epsilon^{-1}\rceil})$ [22]	$O(m^2(\log n + \log w_{\max}))$ [24], $\leqslant 2em^2(1+\ln m+\ln w_{\max})$ [25]
GP: (1+1)-GP	$O\left(\left(\frac{3m\min\{m,n\}}{2}\right)^{\lceil\epsilon^{-1}\rceil}\right)$	$O(mn(\log n + \log w_{\max}))$
GA: GSEMO		$O(mn(n+\log w_{\max}))$ [27]
GP: SMO-GP		$O(mn^2)$

The rest of this paper is organized as follows. Section 2 introduces GP. The running time analysis as well as empirical analysis on the MM and the MST problems is presented in Sections 3 and 4, respectively. Section 5 concludes the paper.

2 Genetic programming

In GPs, solutions are usually represented by the tree data structure. Given a set of functions F (e.g., arithmetic operators) and a set of terminals T (e.g., variables), an internal node of the tree denotes a function in F and a leaf node denotes a terminal in T.

(1+1)-GP, as in Algorithm 1, is a simple GP algorithm. It first initializes a solution (e.g., a random generated tree), then repeatedly generates an offspring solution by mutation, and updates the parent solution if the offspring is better. Note that we consider minimization problems in Algorithm 1; for maximization problems, the condition of the fourth step changes to be "if $f(x') \geqslant f(x)$".

Algorithm 1 (1+1)-GP [11]

```
Input: a solution space $\mathcal{X}$ and an objective function $f$;
Output: a solution $x$;
1: $x \leftarrow$ an initial solution from $\mathcal{X}$;
2: while some criterion is not met do
3:     Create $x'$ by applying mutation to $x$;
4:     if $f(x') \leqslant f(x)$ then
5:         $x \leftarrow x'$;
6:     end if
7: end while
8: return $x$
```

The mutation, as in Definition 1, applies one of the three operators, substitution, insertion, and deletion, uniformly at random and repeats this process k times independently. For (1+1)-GP-single, $k = 1$. The three operators are illustrated in Figure 1. Note that the arity of each function in F studied in this paper is 2. Thus, for simplicity, the three operators are described in such case.

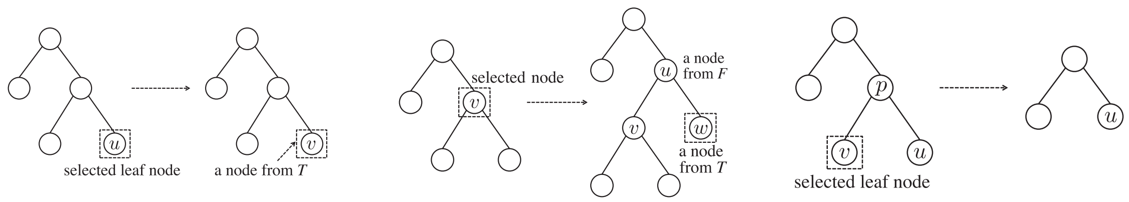
Definition 1 (Mutation)**.** Apply one of the following three operators uniformly at random; this process is repeated independently k times.

[Substitution] Replace a randomly chosen leaf node of the solution with a new node selected uniformly at random from T.

[Insertion] Select a node v of the solution randomly, select a node u from F randomly, and select a node w from T randomly. Replace v with u whose children are v and w, the order of which is random.

[Deletion] Select a leaf node v of the solution randomly, the parent and the sibling of which are p and u, respectively. Replace p with u and delete p and v.

The running time of (1+1)-GP is counted as the number of fitness evaluations until a desired solution is found for the first time, because the fitness evaluation is deemed as the most costly computational process [18, 28]. The desired solutions can be optimal solutions or approximate solutions depending on the application. For approximate optimization (maximization), a $(1 + \epsilon)$-approximate solution x implies that $(1 + \epsilon) \cdot f(x) \geqslant f^*$, where f^* is the maximal objective value.



(a) Substitution (b) Insertion (c) Deletion

Figure 1 Mutation on trees.

SMO-GP is a multi-objective version of (1+1)-GP that simultaneously optimizes two or more objective functions $f_1, \ldots, f_m$. Given a feasible solution space $\mathcal{X}$, the minimum multi-objective optimization is formalized as

$$\arg\min_{x\in\mathcal{X}} \boldsymbol{f}(x) = \arg\min_{x\in\mathcal{X}} \big(f_1(x), ..., f_m(x)\big).$$

Since the objectives are usually conflicted, it is impossible to have one solution that optimizes all the objectives simultaneously. Therefore, multi-objective optimization tries to find a set of solutions according to the Pareto optimality, which uses the *domination* relation between solutions as in Definition 2. The solution set by Pareto optimality is called Pareto set, as in Definition 3.

Definition 2 (Domination)**.** Let $\boldsymbol{f} = (f_1, \ldots, f_m) : \mathcal{X} \to \mathbb{R}^m$ be the objective vector, where $\mathcal{X}$ is the feasible solution space. For two solutions x and $x' \in \mathcal{X}$:
1. *x weakly dominates x' ($x \succeq_{\boldsymbol{f}} x'$)* if $\forall i : f_i(x) \leqslant f_i(x')$;
2. *x dominates x' ($x \succ_{\boldsymbol{f}} x'$)* if $x \succeq_{\boldsymbol{f}} x' \wedge \exists i : f_i(x) < f_i(x')$.

Definition 3 (Pareto Optimality)**.** Let $\boldsymbol{f} : \mathcal{X} \to \mathbb{R}^m$ be the objective vector, where $\mathcal{X}$ is the feasible solution space. A solution x is *Pareto optimal* if there is no other solution in $\mathcal{X}$ that dominates x. A set of solutions is called a *Pareto set* if it contains only Pareto optimal solutions. The collection of objective values of a Pareto set is called a *Pareto front*.

SMO-GP, as in Algorithm 2, is a simple multi-objective GP algorithm that repeatedly generates an offspring solution by mutation and maintains a set of non-dominated solutions created so far. We denote SMO-GP-single as that using the mutation with $k = 1$.

Algorithm 2 SMO-GP [16]

```
Input: a solution space $\mathcal{X}$ and an objective vector $\boldsymbol{f}$;
Output: a set of solutions $P$;
1: $x \leftarrow$ an initial solution from $\mathcal{X}$;
2: while some criterion is not met do
3:     Choose $x$ from $P$ uniformly at random;
4:     Create $x'$ by applying mutation to $x$;
5:     if $\nexists z \in P$ such that $z \succ_{\boldsymbol{f}} x'$ then
6:         $P \leftarrow (P - \{z \in P | x' \succeq_{\boldsymbol{f}} z\}) \cup \{x'\}$;
7:     end if
8: end while
9: return $P$
```

The running time of SMO-GP is counted as the number of calls to $\boldsymbol{f}$ until it finds the Pareto front of the largest Pareto set or called the *optimal Pareto front*. That is, it should find at least one corresponding solution for each element in the optimal Pareto front [29, 30].

The Bloat Problem that the solution complexity grows without increasing the quality is often encountered in GPs [6]. One way to solve this problem is the parsimony approach, which prefers the solution with a smaller complexity when the compared solutions have the same fitness value. The *(1+1)-GP with the parsimony approach* just changes the condition in the fourth step of Algorithm 1 to be

$$\text{“}\textbf{if } \big(f(x') < f(x)\big) \text{ or } \big(f(x') = f(x) \wedge C(x') \leqslant C(x)\big)\text{”},$$

where $C(x)$ is the complexity of the solution x. Another way is the multi-objective approach which introduces $C(x)$ as an auxiliary minimized objective, such that the optimal Pareto front automatically contains different trade-offs between the original objective and the complexity objective.

For the complexity measure $C(x)$ of a tree-structured solution, we use its number of nodes in this paper [16, 17].

3 Analysis on MM problem

3.1 MM Problem

The MM problem can be described as follows. Given an undirected graph $G = (V, E)$ on n vertices and m edges where V and E are the vertex set and edge set, respectively, a matching is a subset E' of the edge set E, such that no two edges in E' share a common vertex. Then, the goal is to find a matching with the largest number of edges. Let the edges be indexed as $\{1, 2, \ldots, m\}$.

The MM problem is one of the limited number of combinatorial problems where the running time of EAs has been analyzed. In [22, 23], a solution is represented as a Boolean string of length m. For a solution $x \in \{0, 1\}^m$, $x_i = 1$ means that the edge i is selected by x. They use the following fitness function for maximization:

$$f(x) = \sum_{i=1}^{m} x_i - c \cdot \sum_{v \in V} p(v, x), \tag{1}$$

where $p(v, x) = \max\{0, d(v, x) - 1\}$, $d(v, x)$ is the degree of the vertex v on the subgraph represented by x, and $c \geqslant m + 1$ is a penalty coefficient which makes any matching have a larger fitness value than any non-matching.

For GP on the MM problem, we use $F = \{J\}$ and $L = \{1, 2, \ldots, m\}$, where J is a joint function which has arity 2 and L contains the m edges. The fitness of a tree-structured solution x can be calculated by the procedure in Definition 4.

Definition 4 (Fitness Calculation). Given a tree-structured solution x, its fitness is calculated as
1. Parse the tree x inorder and add the leaves to a list l.
2. Parse the list l from left to right, for each leaf add the corresponding edge to a set P.
3. Compute the fitness of the graph $G_P = (V, P)$.

In our analysis, we will use the fitness function equivalent to Eq. (1). That is, in the third step of Definition 4, it calculates

$$f(x) = |P| - c \cdot \sum\nolimits_{v \in V} p(v, G_P), \tag{2}$$

where $|P|$ denotes the size of the edge set P, $p(v, G_P) = \max\{0, d(v, G_P) - 1\}$, $d(v, G_P)$ is the degree of the vertex v on the subgraph G_P, and $c \geqslant m + 1$. For example, for the graph G in Figure 2, a MM is an arbitrary edge; when computing the fitness of the solution x, $l = \{1, 2, 1\}$, $P = \{1, 2\}$, and then $f(x) = 2 - c \cdot (0 + 0 + 1) = 2 - c$. Note that the edges $1, 2, 3$ correspond to the edges e_1, e_2, e_3 on the graph G, respectively, and the weight information w_1, w_2, w_3 is not used here, but will be used in the following analysis on the MST problem.

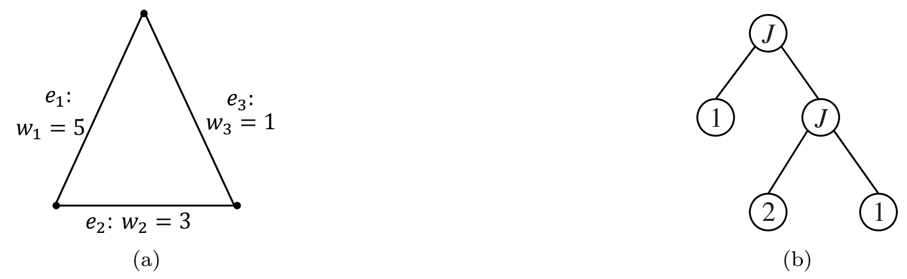


Figure 2 An example of (a) a graph G and (b) a solution x containing the edges $\{1, 2\}$ (i.e., $\{e_1, e_2\}$).

When implementing (1+1)-GP on MM, we use the procedure in Definition 5 to construct the initial tree. Lemma 1 gives properties of the number of leaves of the initial tree by Definition 5, which will be used in the following analysis.

Definition 5 (Initial Tree Construction). Given a set F of functions and a set T of terminals, the tree starts from a root node by randomly selecting a node from $F \cup T$, and then recursively expands a node from F by selecting its children randomly from $F \cup T$, until all the leaf nodes are from T.

Lemma 1. For the initial tree construction in Definition 5, let T_{init} denote the number of leaf nodes of the initial tree. Then, $\mathbb{E}[\![T_{init}]\!] = \frac{m}{m-1}$ and $\mathbb{E}[\![T_{init}^2]\!] \leqslant \frac{m}{m-3}$.

Proof. In our setting, $F = \{J\}$ and $T = \{1, \ldots, m\}$, thus, $|F| = 1$ and $|T| = m$. By the construction procedure in Definition 5, the root node is from T with probability $\frac{m}{m+1}$, and then $T_{init} = 1$; the root node is from F with probability $\frac{1}{m+1}$, and then T_{init} is the sum of the number of leaf nodes of the two subtrees (denoted by T_l and T_r). Thus, we have the recursive equation

$$\mathbb{E}[\![T_{init}]\!] = \frac{m}{m+1} \cdot 1 + \frac{1}{m+1} \cdot (\mathbb{E}[\![T_l]\!] + \mathbb{E}[\![T_r]\!]) = \frac{m}{m+1} + \frac{2}{m+1}\mathbb{E}[\![T_{init}]\!],$$

which results in $\mathbb{E}[\![T_{init}]\!] = \frac{m}{m-1}$. We also have

$$\mathbb{E}[\![T_{init}^2]\!] = \frac{m}{m+1} \cdot 1^2 + \frac{1}{m+1} \cdot \mathbb{E}[\![(T_l + T_r)^2]\!] \leqslant \frac{m}{m+1} + \frac{4}{m+1}\mathbb{E}[\![T_{init}^2]\!],$$

which results in $\mathbb{E}[\![T_{init}^2]\!] \leqslant \frac{m}{m-3}$.

3.2 (1+1)-GP on MM

In this section, we will analyze the running time of (1+1)-GP with the parsimony approach on the MM problem. In the following analysis, we always denote T_{init} as the number of leaves of the initial tree. For a subset P of the whole edge set E, let $|P|$ denote its size, that is, the number of edges contained in P; a vertex is free if it is not the endpoint of any edge in P; an edge $\{v_i, v_j\}$ is free if it does not belong to P and v_i, v_j are free vertexes; and an edge is matching if it belongs to P and $d(v_i, P) = d(v_j, P) = 1$. For a matching M, an augmenting path with respect to M is a path $v_1, \ldots, v_k$ of odd length (i.e., k is even), where the edges $\{v_{2i}, v_{2i+1}\}$ $(1 \leqslant i \leqslant k/2 - 1)$ belong to M and the other edges do not, and also, v_1, v_k are free vertexes.

Theorem 1. The ERT of (1+1)-GP-single with the parsimony approach on the MM problem until finding a $(1+\epsilon)$-optimal matching is $O\left(\left(\frac{3m\min\{m,n\}}{2}\right)^{\lceil \epsilon^{-1} \rceil}\right)$, where $\epsilon > 0$.

Before the proof, we briefly introduce the main proof idea. We first analyze the running time until a matching is found using the multiplicative drift analysis technique as in Lemma 2; then, we derive the running time for a non-MM improving to a $(1+\epsilon)$-optimal matching. In the improving process, we use the property that swapping the edges of an augmenting path of a non-MM can increase the number of matching edges, which has been applied for analyzing (1+1)-EA solving the MM problem in [22]. An upper bound on the length of an augmenting path in Lemma 3 will be used in the following proof.

Lemma 2 (Multiplicative Drift Analysis [25]). Let $S \subseteq \mathbb{R}$ be a finite set of positive numbers with minimum $s_{\min}$. Let $\{X^t\}_{t \in \mathbb{N}}$ be a sequence of random variables over $S \cup \{0\}$. Let T be the random variable that denotes the first point in time $t \in \mathbb{N}$ for which $X^t = 0$. Suppose that there exists a real number $\delta > 0$ such that

$$\mathbb{E}[\![X^t - X^{t+1} \mid X^t = s]\!] \geqslant \delta s$$

holds for all $s \in S$ with $Pr[X^t = s] > 0$. Then, for all $s_0 \in S$ with $Pr[X^0 = s_0] > 0$, we have

$$\mathbb{E}[\![T \mid X^0 = s_0]\!] \leqslant \frac{1 + \ln(s_0/s_{min})}{\delta}.$$

Lemma 3 ([21]). Let $G = (V, E)$ be a graph, M a non-MM, and M^* a MM. Then, there exists an augmenting path with respect to M whose length is bounded from above by $2\lfloor |M|/(|M^*| - |M|) \rfloor + 1$.

Proof of Theorem 1. We consider two phases:
• *phase 1*: starts after initialization and finishes until a solution which represents a matching M and has exactly $|M|$ leaves is found.
• *phase 2*: starts after *phase 1* and finishes until a solution which represents a $(1+\epsilon)$-optimal matching is found.
We first analyze each phase i and derive an upper bound E_i on the ERT. Then, by summing them up, we get an upper bound E_1+E_2 on the ERT of the whole optimization process.

In *phase 1*, let x_t $(t \geqslant 0)$ denote the solution after t generations. Let $X^t=\sum_v p(v,x_t)+\sum_{e\in x_t}(N(e)-1)$, where $p(v,x_t)=p(v,G_P)$ as in Eq. (2), $G_P=(V,P)$ is the graph generated in Definition 4 when calculating the fitness of x_t, $e\in x_t$ means $e\in P$, and $N(e)$ denotes the number of occurrences of edge e in the leaves of the solution x_t. We use multiplicative drift analysis (i.e., Lemma 2) to get the running time until $X^t=0$ (i.e., a matching is found and there is no duplicate edge in the solution). We are to analyze $\mathbb{E}[\![X^t-X^{t+1}\mid X^t]\!]$.

We first show that X^t is non-increasing with t. It is easy to see that $\sum_v p(v,x_t)$ never increases, since this term dominates the fitness function f (i.e., Eq. (2)) and f never decreases. Then, we consider three possible operators in one mutation step.
(1) by deletion, for any $e\in x_t$, $N(e)$ obviously cannot increase, thus, X^t cannot increase.
(2) by insertion, only inserting free edges can be accepted. This will add a new edge e with $N(e)=1$, while the term $\sum_{e\in x_t}(N(e)-1)$ does not increase. Thus, X^t cannot increase.
(3) by substitution, we can view it as deletion first and then insertion. If it deletes an edge e with $N(e)>1$ which decreases $\sum_{e\in x_t}(N(e)-1)$ by 1, any following insertion cannot increase X^t, because it can increase $\sum_{e\in x_t}(N(e)-1)$ by at most 1. The only possible way to increase $\sum_{e\in x_t}(N(e)-1)$ is to delete an edge e with $N(e)=1$ and insert an edge e' with $N(e')\geqslant 1$, which will increase $\sum_{e\in x_t}(N(e)-1)$ by 1. Then, we consider the effect on $\sum_v p(v,x_t)$ by this event. Inserting an edge e' with $N(e')\geqslant 1$ will not affect $\sum_v p(v,x_t)$. For deleting an edge e with $N(e)=1$, if it is a matching edge, $\sum_v p(v,x_t)$ will not be affected while $|P|$ decreases by 1, thus the fitness Eq. (2) decreases and the offspring will be rejected; otherwise, the deleted edge e shares endpoints with other edges, and its deletion will decrease $\sum_v p(v,x_t)$ by at least 1, thus, X^t will not increase.

Then, we are to show that in every mutation step, there exist at least $\lceil X^t/4\rceil$ leaves whose deletion will be accepted and will decrease X^t by at least 1 (i.e., $X^t-X^{t+1}\geqslant 1$). If $\sum_{e\in x_t}(N(e)-1)>\lfloor X^t/2\rfloor$, deleting one of these duplicate edges will not affect the fitness while decreasing the complexity of the solution; thus X^t decreases by 1. If $\sum_{e\in x_t}(N(e)-1)\leqslant\lfloor X^t/2\rfloor$, that is, $\sum_v p(v,x_t)\geqslant\lceil X^t/2\rceil$, there are at least $\lceil\sum_v p(v,x_t)/2\rceil\geqslant\lceil X^t/4\rceil$ edges contributing to the value $\sum_v p(v,x_t)$. For one of these edges e, if $N(e)=1$, deleting it will decrease $\sum_v p(v,x_t)$ by at least 1; if $N(e)>1$, deleting it will not affect $\sum_v p(v,x_t)$ while decreasing $\sum_{e\in x_t}(N(e)-1)$ by 1. Thus, our claim holds. Let L_t denote the number of leaves of the solution x_t. Then, the probability of decreasing X^t in one step is at least $\frac{1}{3}\cdot\frac{X^t/4}{L_t}$, where $\frac{1}{3}$ is the probability of applying deletion in mutation.

Based on the above analysis of X^t, we can derive

$$\mathbb{E}[\![X^t-X^{t+1}\mid X^t]\!]\geqslant X^t/12L_t.$$

Then, we are to give an upper bound on L_t. Note that $\sum_{e\in x_t}(N(e)-1)$ is the number of duplicate edges contained in the leaves of x_t, and $\sum_v p(v,x_t)$ gives the number of edges whose deletion on G_P can generate a matching. Because the number of edges of a matching is not larger than $\min\{m,n\}$, the number of leaves is upper bounded by $X^t+\min\{m,n\}$, that is, $L_t\leqslant X^t+\min\{m,n\}$. It is easy to see that $X^0\leqslant\sum_{e\in x_0}(N(e)-1)+2(T_{init}-\sum_{e\in x_0}(N(e)-1))\leqslant 2T_{init}$. Also, we have shown that X^t is non-increasing. Thus, $X^t\leqslant 2T_{init}$, which leads to $L_t\leqslant 2T_{init}+\min\{m,n\}$. Then, we can get

$$\mathbb{E}[\![X^t-X^{t+1}\mid X^t]\!]\geqslant X^t/12(2T_{init}+\min\{m,n\}).$$

By Lemma 2 and $\min\{X^t>0\}=1$, we derive $\mathbb{E}[\![T\mid X^0]\!]\leqslant 12(2T_{init}+\min\{m,n\})(1+\ln(X^0))$. Due to $X^0\leqslant 2T_{init}$, we have

$$E_1\in O((T_{init}+\min\{m,n\})\ln T_{init}).$$

In *phase 2*, the solution will always be a matching, because a non-matching has a smaller fitness than a matching, and also, the solution has no duplicate edges, because inserting a duplicate edge will increase the complexity while not affecting the fitness, and substituting an existing edge with a duplicate edge will decrease the fitness by 1. Then, it is easy to see that the number of leaves of the solution in this phase is always not larger than $\min\{m,n\}$, which implies that the probability of a specific substitution in a mutation step is at least $\frac{1}{3}\cdot\frac{1}{m\cdot\min\{m,n\}}$, where $\frac{1}{3}$ is the probability of applying substitution in mutation, $\frac{1}{\min\{m,n\}}$ is a lower bound for the probability of deleting a specific leaf of the solution, and $\frac{1}{m}$ is the probability of selecting a specific edge from T for insertion.

Let M be the matching represented by the current solution and M^* be a MM. By Lemma 3, there exists an augmenting path l with length bounded above by $2\lfloor |M|/(|M^*|-|M|)\rfloor+1$. Since we are to find a $(1+\epsilon)$-optimal matching, we have $(1+\epsilon)|M|<|M^*|$ before this phase finishes. Thus, $|l|\leqslant 2\lceil\epsilon^{-1}\rceil-1$, where $|l|$ denotes the length of l. Note that $|l|$ is odd. To improve the current matching (i.e., to increase $|M|$), we can delete all the edges on l which belong to M and insert all the edges on l which do not belong to M. Such behavior will increase $|M|$ by 1, by the definition of augmenting path. For achieving it, we can exchange the first or last two edges of l by substitution in one step, then repeat this process $\lfloor |l|/2\rfloor$ times, and finally insert the remaining edge on l. Note that each of the $\lfloor |l|/2\rfloor$ substitution is accepted since it does not affect the fitness and the complexity; and the last insertion is also accepted since it increases the fitness by 1. We call such a procedure a successful phase, the probability of which is at least $\left(2\cdot\frac{1}{3m\cdot\min\{m,n\}}\right)^{\lfloor |l|/2\rfloor}\cdot\frac{1}{3m}\in\Omega(2^{\lfloor |l|/2\rfloor}/((3m)^{\lceil |l|/2\rceil}(\min\{m,n\})^{\lfloor |l|/2\rfloor}))$, where the factor 2 is because we have two choices for substitution, that is, exchanging the first two edges of an augmenting path or the last two. Thus, the expected number of phases until a successful phase is $O\left((3m)^{\lceil |l|/2\rceil}\left(\frac{\min\{m,n\}}{2}\right)^{\lfloor |l|/2\rfloor}\right)$, which leads to an obvious upper bound $O\left(\lceil |l|/2\rceil\cdot(3m)^{\lceil |l|/2\rceil}\left(\frac{\min\{m,n\}}{2}\right)^{\lfloor |l|/2\rfloor}\right)$ on the expected number of steps until success. We are then to show a tighter upper bound by making full use of the fact that these $O\left((3m)^{\lceil |l|/2\rceil}\left(\frac{\min\{m,n\}}{2}\right)^{\lfloor |l|/2\rfloor}\right)$ phases are unsuccessful. It is easy to see that a phase can continue successfully with probability at most $\frac{1}{3m}$. Thus, the expected number of steps for an unsuccessful phase is $O(1)$. Then, we can derive that the expected number of steps until success is $O\left((3m)^{\lceil |l|/2\rceil}\left(\frac{\min\{m,n\}}{2}\right)^{\lfloor |l|/2\rfloor}+\lceil |l|/2\rceil\right)\in O\left((3m)^{\lceil |l|/2\rceil}\left(\frac{\min\{m,n\}}{2}\right)^{\lfloor |l|/2\rfloor}\right)$, where the term $\lceil |l|/2\rceil$ after '+' is the length of the final successful phase. The rigorous proof for this tight bound can be found in Appendix C of [31]. Since $|M^*|\leqslant\min\{m,n\}$, we need at most $\min\{m,n\}$ successes to find a $(1+\epsilon)$-optimal matching. Thus, the expected time for this phase is at most

$$E_2\in O\left((3m)^{\lceil |l|/2\rceil}\left(\frac{\min\{m,n\}}{2}\right)^{\lceil |l|/2\rceil}\right)\in O\left(\left(\frac{3m\min\{m,n\}}{2}\right)^{\lceil\epsilon^{-1}\rceil}\right).$$

Thus, the ERT of the whole process starting from an initial tree is at most

$$E_1+E_2\in O\left((T_{init}+\min\{m,n\})\ln T_{init}+\left(\frac{3m\min\{m,n\}}{2}\right)^{\lceil\epsilon^{-1}\rceil}\right),$$

which is a random variable depending on T_{init}. Because the initial tree is constructed by Definition 5, we have $\mathbb{E}[\![T_{init}]\!]=\frac{m}{m-1}$ by Lemma 1; $\mathbb{E}[\![\ln T_{init}]\!]\leqslant\ln\mathbb{E}[\![T_{init}]\!]=\ln\frac{m}{m-1}$ by Jensen's inequality and the concavity of the function $\ln x$; and $\mathbb{E}[\![T_{init}\ln T_{init}]\!]\leqslant\mathbb{E}[\![T_{init}^2]\!]\leqslant\frac{m}{m-3}$ by Lemma 1. Thus, the ERT over all possible initial trees is $O\left(\left(\frac{3m\min\{m,n\}}{2}\right)^{\lceil\epsilon^{-1}\rceil}\right)$. □

Thus, we have proved that (1+1)-GP solves the MM problem in $O\left(\left(\frac{3m\min\{m,n\}}{2}\right)^{\lceil\epsilon^{-1}\rceil}\right)$ ERT. Compared with the ERT $O(m^{2\lceil\epsilon^{-1}\rceil})$ of (1+1)-EA on MM reported in literature [21, 22], (1+1)-GP has a better upper bound.

Through comparing our analysis with the analysis of (1+1)-EA in [21], we find that the efficiency of GP is due to the larger probability of exchanging two edges on an augmenting path in the process of improving a non-MM. For GA, exchanging two edges is by flipping the bits on the corresponding two positions which occurs with probability $\Theta(\frac{1}{m^2})$; while for GP, exchanging two edges is by substitution, which deletes a specific edge from the current solution with probability at least $\frac{1}{\min\{m,n\}}$ and then inserts a specific edge from T with probability $\frac{1}{m}$. The large probability of deletion for GP is because in *phase 2* of the optimization, the evolved solution always represents a matching and has no duplicate edges, which makes its number of leaves always not larger than $\min\{m, n\}$. Thus, our analysis discloses that variable solution structure might be helpful for evolutionary optimization if the solution complexity can be controlled well.

3.3 Empirical studies

When comparing GPs with GAs on the MM problem, the running time bounds might not be tight, which makes the comparison not quite strict. To rigorously show that (1+1)-GP is better than (1+1)-EA on the MM problem, we need to prove that the upper bound of (1+1)-GP is smaller than the lower bound of (1+1)-EA. However, the lower bound analysis is quite difficult, and there has not existed any general lower bound for (1+1)-EA on MM. Therefore, we conduct experiments to supplement the theoretical analysis. We compare these algorithms on the graphs with the number of edges $m \in \Theta(n)$, $\Theta(n\sqrt{n})$, and $\Theta(n^2)$, respectively, which are briefly called as sparse, moderate, and dense graphs, respectively. Let $v_1, v_2, \ldots, v_n$ denote the n nodes.

sparse graph: we use cyclic graph where v_1 is connected with v_n and v_2, v_i $(1 < i < n)$ is connected with v_{i-1} and v_{i+1}, and v_n is connected with v_{n-1} and v_1. Thus, $m = n$.

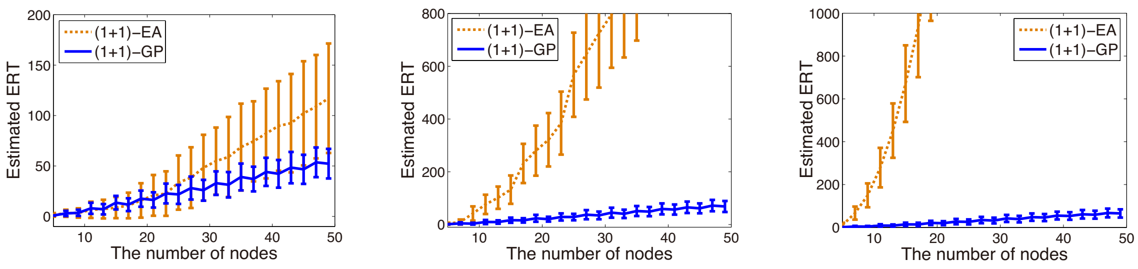
moderate graph: we use the graph where v_i is connected with $v_{i+1}, v_{i+2}, \ldots, v_{i+\lfloor\sqrt{n}\rfloor}$ for $1 \leqslant i \leqslant n - \lfloor\sqrt{n}\rfloor$. Thus, $m = (n - \lfloor\sqrt{n}\rfloor)\lfloor\sqrt{n}\rfloor$.

dense graph: we use complete graph where each node is connected with all the other nodes. Thus, $m = n(n-1)/2$.

For each kind of graph, it is easy to see that the cardinality of a MM is $\lfloor\frac{n}{2}\rfloor$. The range of the number of nodes n is set to be $[5, 50]$. On each size of n, we repeat independent runs for 1000 times, and then the average running time is recorded as an estimation of the ERT; we also record the standard deviation of the running time on each size as the bar in the figures. For the approximation, we set ϵ to be $0.1, 0.5, 1$, respectively.

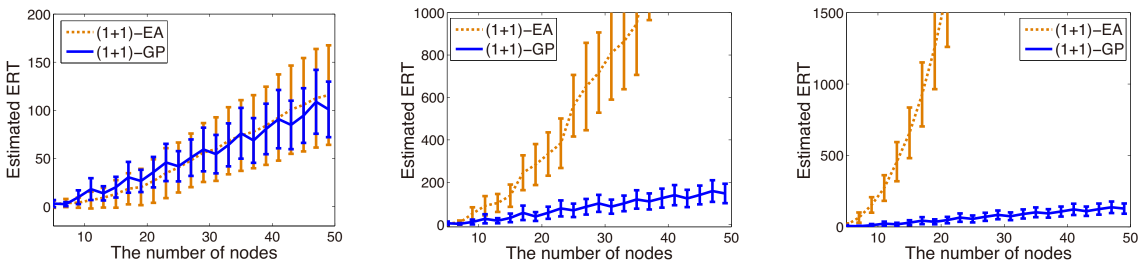
The experimental results for different ϵ values are plotted in Figures 3–5, respectively. We can observe that the curve of (1+1)-GP is much lower than that of (1+1)-EA in Figures 3–5(b–c); their curves are a little close in Figures 3–5(a). From the comparison of the running time bounds as in the second column of Table 1, we can derive that (1+1)-GP performs as similar as (1+1)-EA for sparse graph ($m \in \Theta(n)$), while (1+1)-GP is better for moderate ($m \in \Theta(n\sqrt{n})$) and dense ($m \in \Theta(n^2)$) graphs. Thus, the empirical results are consistent with the theoretical results.

From Figure 5(a–b), it can be observed that there are two peaks at $n = 21, 43$. We are then to give some explanation. Note that the size of a MM is $\lfloor\frac{n}{2}\rfloor$. The $(1+\epsilon)$-optimal matching that we are to find has to contain at least $NE = \lceil\lfloor\frac{n}{2}\rfloor/(1+\epsilon)\rceil$ number of edges. For $\epsilon = 0.1$, when $n \in [0, 21]$, $NE = \lfloor\frac{n}{2}\rfloor$; when $n \in [22, 43]$, $NE = \lfloor\frac{n}{2}\rfloor - 1$; when $n \in [44, 65]$, $NE = \lfloor\frac{n}{2}\rfloor - 2$; in summary, when $n \in [22k, 22(k+1) - 1]$ for any $k \geqslant 0$, $NE = \lfloor\frac{n}{2}\rfloor - k$. By the relation between NE and $\lfloor\frac{n}{2}\rfloor$, we can expect that the running time increases with n when $n \in [22k, 22(k+1) - 1]$, because we are always to find a matching with k edges less than the MM and the problem becomes harder as the number of nodes increases; the running time has a sudden decrease at $n = 22(k+1)$ because we are currently to find a matching with $k+1$ edges less than the MM and the problem becomes easier than that for $n = 22(k+1) - 1$. Thus, the empirical observation from Figure 5(a–b) is consistent with our analysis. However, for other figures, we cannot observe the obvious sudden decrease in the running time, which may be because the problem hardness decrease due to the increase of $\lfloor\frac{n}{2}\rfloor - NE$ is smaller than the problem hardness increase due to the increase in the number of nodes n.



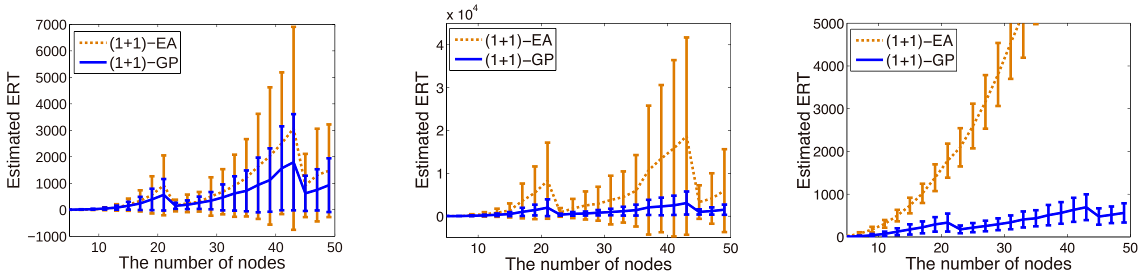
(a) Sparse graph (b) Moderate graph (c) Dense graph

Figure 3 Estimated ERT comparison on the MM problem for different graphs with $\epsilon = 1$.



(a) Sparse graph (b) Moderate graph (c) Dense graph

Figure 4 Estimated ERT comparison on the MM problem for different graphs with $\epsilon = 0.5$.



(a) Sparse graph (b) Moderate graph (c) Dense graph

Figure 5 Estimated ERT comparison on the MM problem for different graphs with $\epsilon = 0.1$.

4 Analysis on MST problem

4.1 MST problem

The MST problem can be described as follows. Given an undirected connected graph $G = (V, E)$ on n vertices and m edges where V and E are the vertex set and edge set, respectively, the goal is to find a connected graph $G' = (V, E' \subseteq E)$ with the minimal sum of edge weights. Let the edges be indexed as $\{1, 2, \ldots, m\}$ with non-negative weights $\{w_1, w_2, \ldots, w_m\}$. Let $w_{\max}$ denote the maximal weight, that is, $w_{\max} = \max\{w_i \mid 1 \leqslant i \leqslant m\}$.

For EAs solving the MST problem, Neumann and Wegener [24] represent a solution x by a Boolean string of length m, that is, $x \in \{0, 1\}^m$, where $x_i = 1$ means that the edge i is selected by x. They use the following fitness function for minimization:

$$f(x) =(c(x) - 1)w_{ub}^2 + \left(\sum\nolimits_{i=1}^{m} x_i - n + 1\right) w_{ub} + \sum\nolimits_{i=1}^{m} x_i w_i, \tag{3}$$

where $c(x)$ is the number of connected components by the edges in x, and $w_{ub} = n^2 w_{\max}$. Note that in this fitness function, the first term $(c(x) - 1)w_{ub}^2$ makes a subgraph with fewer connected components better, the second term $(\sum_{i=1}^{m} x_i - n + 1)w_{ub}$ makes a connected subgraph with fewer edges better, and the last term $\sum_{i=1}^{m} x_i w_i$ makes a spanning tree with a smaller weight better.

For GP on the MST problem, we use $F = \{J\}$ and $L = \{1, 2, \ldots, m\}$, where J is a joint function with arity 2 and L contains the m edges. The fitness of a tree-structured solution x can be calculated by the procedure in Definition 4. In our analysis, we will use the fitness function equivalent to Eq. (3). That is, in the third step of Definition 4, it calculates

$$W(x) = (c(G_P) - 1)w_{ub}^2 + (|P| - (n-1))w_{ub} + \sum\nolimits_{i \in P} w_i, \tag{4}$$

where $c(G_P)$ denotes the number of connected components in graph G_P, and $|P|$ denotes the size of the edge set P. For example, when computing the fitness of the solution x for the graph G in Figure 2, $l = \{1, 2, 1\}$, $P = \{1, 2\}$, and then $W(x) = (1-1) \cdot w_{ub}^2 + (2-2) \cdot w_{ub} + w_1 + w_2 = 8$.

When implementing (1+1)-GP and SMO-GP on MST, we use the procedure in Definition 5 to construct the initial tree.

4.2 (1+1)-GP on MST

In this section, we will analyze the running time of (1+1)-GP with the parsimony approach on the MST problem. We assume that all the edge weights are integers, as in the previous analysis of EAs on MST [24, 25, 27]. In the following analysis, we will not distinguish a solution and the subgraph G_P it represents for convenience.

Theorem 2. The ERT of (1+1)-GP-single with the parsimony approach on the MST problem is $O(mn(\log n + \log w_{\max}))$.

Before the proof, we first give a property on local changes of spanning trees as in Lemma 4.

Lemma 4 ([24]). Let M describe a non-MST. Then, there exist $k \in [1, n-1]$ different two-edge exchanges which delete one edge in M and insert one edge which has not been in M, such that each two-edge exchange generates a spanning tree with a smaller weight than M, and the average weight decrease of these k exchanges is $(W_M - W_{opt})/k$, where W_M and W_{opt} are the weights of M and the MST, respectively.

Proof of Theorem 2. Inspired from the analysis of (1+1)-EA on the MST problem in [24], we divide the optimization into three phases as follows:

• *phase 1*: starts after initialization and finishes until a solution representing a connected subgraph is found.

• *phase 2*: starts after *phase 1* and finishes until a solution with $n-1$ leaves which represents a spanning tree is found.

• *phase 3*: starts after *phase 2* and finishes until a solution representing a MST is found.

We first analyze each phase i and derive an upper bound E_i on the ERT. Then, by summing them up, we get an upper bound $E_1 + E_2 + E_3$ on the ERT of the whole optimization process.

In *phase 1*, let c denote the number of connected components of the current solution. Then, c cannot increase, because a solution with more connected components has a larger fitness value by the definition of $W(x)$ (i.e., Eq. (4)). For a subgraph with c connected components, there must exist at least $c-1$ edges whose insertion decreases the number of connected components by 1, since the original graph is connected. Thus, the probability of decreasing c by 1 in one step for (1+1)-GP-single is at least $\frac{1}{3} \cdot \frac{c-1}{m}$, where $\frac{1}{3}$ is the probability of applying insertion in mutation, and $\frac{c-1}{m}$ is the probability of selecting one of those $c-1$ edges from T for insertion. Then, the expected number of steps for decreasing c by 1 is at most $\frac{3m}{c-1}$. By $c \leqslant n$, we get that the ERT until a connected subgraph is found (i.e., $c = 1$) is at most

$$E_1 = \sum\nolimits_{c=2}^{n} \frac{3m}{c-1} \in O(m \log n).$$

In *phase 2*, let l denote the number of leaves of the current solution. Note that in this phase, the solution is always connected, because a non-connected subgraph has a larger fitness than a connected subgraph by Eq. (4). It is easy to see that l cannot increase, since the insertion of a new leaf u will increase the fitness $W(x)$ if u represents a new edge, or will keep $W(x)$ unchanged but increase the complexity $C(x)$ if u represents an edge which has been in the solution. For a solution with l leaves which represents a connected subgraph, there exist at least $l-(n-1)$ leaves whose deletion decreases l by 1 and keeps the subgraph connected, since it contains at least one spanning tree and a spanning tree contains exactly $n-1$ number of edges. Thus, the probability of decreasing l by 1 in one step is at least $\frac{1}{3}\cdot\frac{l-n+1}{l}$, where $\frac{1}{3}$ is the probability of applying deletion in mutation, and $\frac{l-n+1}{l}$ is the probability of selecting one of those $l-n+1$ leaves from all the l leaves for deletion. Let l_{init} be the number of leaves of the solution when this phase starts. Thus, the ERT until l decreases to $n-1$ is at most $\sum_{l=n}^{l_{init}}\frac{3l}{l-n+1}$. Note that $l=n-1$ implies that the current solution is a spanning tree, since the solution is always connected. Thus,

$$E_2=\sum\nolimits_{l=n}^{l_{init}}\frac{3l}{l-n+1}.$$

Then, we are to get an upper bound on l_{init}. The number of leaves can only increase by insertion. In the first phase of finding a connected subgraph, insertion can be accepted only if it can decrease the number of connected components by 1; otherwise, it will be rejected since it will either increase the number of edges or keep the edges but increase the complexity. Since the number of connected components can decrease at most $n-1$ times, insertion can be accepted by at most $n-1$ times in *phase 1*. Thus, $l_{init}\leqslant T_{init}+n-1$. Then,

$$E_2\leqslant\sum_{l=n}^{T_{init}+n-1}\frac{3l}{l-n+1}\in O(T_{init}+n\log T_{init}).$$

In *phase 3*, the solution will always be a spanning tree, since a non-spanning tree has a larger fitness than a spanning tree. Note that, after *phase 2*, the spanning tree found has exactly $n-1$ leaves. Then, in this phase, insertion and deletion will not be accepted, because insertion will lead to more edges or the same edges with a larger complexity and deletion will lead to more connected components. Thus, the number of leaves of the solution during this phase is always $n-1$.

We are then to prove the ERT until finding a MST using Lemma 2. Let $X^t=W(x_t)-W_{opt}$, where x_t is the solution after t generations in this phase. Then, $\mathbb{E}[\![X^t-X^{t+1}\mid X^t]\!]=W(x_t)-\mathbb{E}[\![W(x_{t+1})\mid x_t]\!]$. $W(x_t)$ is non-increasing with t, since a spanning tree with a larger weight will be rejected. From Lemma 4, we know that for a solution x which represents a spanning tree, there exist k accepted substitutions whose average weight decrease is $\frac{W(x)-W_{opt}}{k}$ for some $k\in[1,n-1]$. Note that the probability of a specific substitution is $\frac{1}{3}\cdot\frac{1}{m(n-1)}$, where $\frac{1}{3}$ is the probability of applying substitution in mutation, $\frac{1}{n-1}$ is the probability of deleting a specific edge from the $n-1$ leaves, and $\frac{1}{m}$ is the probability of selecting one of the m edges from T for insertion. Thus, we have

$$\mathbb{E}[\![X^t-X^{t+1}\mid X^t]\!]\geqslant\frac{k}{3m(n-1)}\cdot\frac{W(x_t)-W_{opt}}{k}=\frac{1}{3m(n-1)}X^t.$$

By Lemma 2 and $\min\{X^t>0\}\geqslant 1$, we derive $\mathbb{E}[\![T\mid X^0]\!]\leqslant 3m(n-1)(1+\ln(X^0))$. Since this phase starts from a spanning tree, $X^0\leqslant nw_{\max}$. Thus, we have

$$E_3\in O(mn(\log n+\log w_{\max})).$$

Using $\mathbb{E}[\![T_{init}]\!]=\frac{m}{m-1}$ in Lemma 1, we can get that the ERT of the whole process is at most

$$\mathbb{E}[\![E_1+E_2+E_3]\!]\in O(mn(\log n+\log w_{\max})).$$

□

Thus, we have proved that (1+1)-GP solves the MST problem in $O(mn(\log n+\log w_{\max}))$ ERT. Compared with the ERT $O(m^2(\log n+\log w_{\max}))$ of (1+1)-EA on the MST problem reported in literature [24, 25], (1+1)-GP has a better upper bound. In [24], it was also proved that (1+1)-EA needs

$\Theta(n^4 \log n)$ ERT for an example graph with $m \in \Theta(n^2)$ and $w_{\max} = n^2$. From Theorem 2, we can conclude that (1+1)-GP is rigorously better than (1+1)-EA on this example, because the ERT of (1+1)-GP is upper bounded by $O(n^3 \log n)$.

By comparing our analysis with the analysis of (1+1)-EA in [24], we can find a similar reason why GP is more efficient as that we have found from the analysis on the MM problem. It is because in the process of improving a non-MST (i.e., *phase 3*), (1+1)-EA needs to flip two specific bits for a good substitution, which happens with probability $\Theta(\frac{1}{m^2})$; while (1+1)-GP can perform such a substitution by deleting a specific edge from the current solution with probability $\frac{1}{n-1}$ and then inserting a specific edge from T with probability $\frac{1}{m}$. The large probability of deletion for (1+1)-GP is since the solution in *phase 3* always has exactly $n-1$ leaves. Thus, the analysis on the MST problem also discloses that variable solution structure might be helpful if the solution complexity is properly controlled.

4.3 SMO-GP on MST

It has been shown that the MST problem can be solved in $O(mn(n+\log w_{\max}))$ ERT by the multi-objective reformulation [27], which transforms the original problem to a two-objective problem by introducing an auxiliary objective and employs a multi-objective GA (i.e., GSEMO, a multi-objective version of (1+1)-EA) to solve the transformed problem. In this section, we analyze SMO-GP solving the multi-objective reformulated MST problem MO-MST by treating the complexity of a solution as the auxiliary objective. Note that the analysis of SMO-GP on MO-MST does not require that the edge weights are integers.

Definition 6 (MO-MST)**.** The two-objective MST problem MO-MST is defined as follows:

$$\text{MO-MST}(x) = \big(W(x), C(x)\big),$$

where $C(x)$ is the complexity of the solution x, that is, the number of nodes in the GP-tree of x.

Thus, for MO-MST, in the third step of Definition 4, besides computing $W(x)$, it also needs to compute the complexity $C(x) = 2 \cdot |l| - 1$. For example, the solution x in Figure 2 has the complexity value $C(x) = 2 \cdot 3 - 1 = 5$.

For a solution x with i ($0 \leqslant i \leqslant n-1$) leaves, the subgraph represented by x contains at most i edges, thus contains at least $(n-i)$ connected components. Since the number of connected components dominates the fitness function $W(x)$ (i.e., Eq. (4)), this solution will have the minimum $W(x)$ value (denoted as W_i), when the subgraph it represents has $n-i$ connected components and its weight is the minimum among all the subgraphs with $n-i$ connected components. That is, $W_i = (n-i-1)w_{ub}^2 + (i-(n-1))w_{ub}+$ the minimum weight among all the subgraphs with $n-i$ connected components. Specifically, for a solution with $(n-1)$ leaves, the minimum $W(x)$ value (i.e., W_{n-1}) is just the weight of a MST, which is also the minimum $W(x)$ value for all solutions. This implies that such a solution will dominate any solution with more than $(n-1)$ leaves, because its complexity $C(x)$ is smaller and its $W(x)$ value is not larger. Note that W_i decreases as i increases.

Based on the above analysis, the optimal Pareto front of MO-MST is $\{(W_0, 0)\} \cup \{(W_i, 2i-1) \mid 1 \leqslant i \leqslant n-1\}$. For $(W_0, 0)$, the corresponding solution is the empty tree. For $(W_i, 2i-1)$, its corresponding solution is a tree with i leaves which constitutes a subgraph with the minimum weight among all the subgraphs with $n-i$ connected components. We can see that the Pareto optimal solution with objective value $(W_{n-1}, 2n-3)$ represents a MST.

Theorem 3. The ERT of SMO-GP-single on the MO-MST problem is $O(mn^2)$.

Before the proof, we first give an upper bound on the population size during the evolutionary process, as in Lemma 5.

Lemma 5. For SMO-GP-single on the MO-MST problem, the population size is upper bounded by $T_{init} + n$.

Proof. Let $c(x)$ denote the number of connected components of a solution x. Denote x_t^* as the solution with the maximal number of nodes (i.e., the maximal $C(x)$ value) in the tth population (i.e., the population after t iterations). We are first to show that $c(x_t^*)$ is non-increasing with t (i.e., $c(x_t^*) \geqslant c(x_{t+1}^*)$)

by considering two possible cases of x^*_{t+1}. Note that the population of SMO-GP always consists of non-dominated solutions.

(1) If x^*_{t+1} is the generated offspring in the $(t+1)$th iteration, x^*_t cannot dominate x^*_{t+1}, otherwise, x^*_{t+1} will be discarded in the updating process of SMO-GP. Then, there are two possible situations: x^*_{t+1} weakly dominates x^*_t, or x^*_{t+1} and x^*_t are incomparable. In the first situation, we obviously have $W(x^*_{t+1}) \leqslant W(x^*_t)$. In the second situation, x^*_t will appear in the $(t+1)$th population, then $C(x^*_{t+1}) > C(x^*_t)$ as x^*_{t+1} has the maximal $C(x)$ value in the $(t+1)$th population; thus $W(x^*_{t+1}) < W(x^*_t)$ as they are incomparable. Then, $c(x^*_{t+1}) \leqslant c(x^*_t)$ can be easily derived from $W(x^*_{t+1}) \leqslant W(x^*_t)$.

(2) x^*_{t+1} is a solution which has been in the tth population; we are then to show that such a case is impossible. Assume that this case holds. It implies that x^*_t must be deleted, otherwise, x^*_{t+1} cannot be the solution with the maximal number of nodes in the $(t+1)$th population since $C(x^*_t) > C(x^*_{t+1})$. Based on the reproduction behavior of SMO-GP and the fact that x^*_t is deleted, it must hold that the generated offspring (denoted as x) in the $(t+1)$th iteration *weakly dominates* x^*_t, and x has been included into the $(t+1)$th population. Since x and x^*_{t+1} appear simultaneously in the $(t+1)$th population, they are incomparable, which implies that $W(x) > W(x^*_{t+1})$ by $C(x) < C(x^*_{t+1})$. Because x^*_{t+1} appears in the tth population, x^*_{t+1} and x^*_t are incomparable, which implies that $W(x^*_{t+1}) > W(x^*_t)$ by $C(x^*_t) > C(x^*_{t+1})$. Thus, we can get $W(x) > W(x^*_t)$, which, however, contradicts with that x *weakly dominates* x^*_t.

We are then to analyze the increase in $C(x^*_t)$. The number of nodes of a solution can only increase by insertion. An offspring solution x' generated by insertion on x may be accepted only when the number of connected components decreases by 1; otherwise, $W(x') \geqslant W(x) \wedge C(x') > C(x)$, and then x' will be rejected. Note that in the population, every solution has a unique number of nodes, because they are non-dominated. Thus, $C(x^*_{t+1}) > C(x^*_t)$ may happen only when in the reproduction, it applies insertion on x^*_t and insertion decreases $c(x^*_t)$ by 1. Since $c(x^*_t) \leqslant n$ and it cannot increase with t, $c(x^*_t)$ can decrease by at most $(n-1)$ times, which implies that $C(x^*_t)$ can increase by at most $(n-1)$ times.

Note that insertion just increases the number of leaves by 1. Thus, the number of leaves of the solutions in the population is always not larger than $T_{init} + n - 1$, which implies that there are at most $T_{init} + n$ possible second objective (i.e., $C(x)$) values. Because for each $C(x)$ value, there exists at most one corresponding solution in the population, the population size is always not larger than $T_{init} + n$.

Proof of Theorem 3. Inspired from the analysis of GSEMO solving the multi-objective formulation of the MST problem in [27], we divide the optimization into two phases:

• *phase 1*: starts after initialization and finishes until the empty graph (i.e., the objective vector $(W_0, 0)$) is found.

• *phase 2*: starts after *phase 1* and finishes until the optimal Pareto front is found.

We first derive a running time upper bound E_i for each phase i, and then sum them up to get an upper bound $E_1 + E_2$ on the ERT of the whole evolutionary process.

In *phase 1*, let l denote the minimal number of leaves of the solutions in the current population, and let x^* denote the corresponding solution with l leaves. Then, l will never increase, since a solution with more leaves cannot weakly dominate a solution with fewer leaves. By applying deletion on x^*, the offspring will always be accepted, since its second objective (i.e., complexity) value becomes the smallest and then it is not dominated by any solution in the current population; and thus l decreases by 1. After applying such step l times, the empty graph will be found. The probability of applying deletion on x^* is at least $\frac{1}{T_{init}+n} \cdot \frac{1}{3}$, because the probability of selecting a specific solution for mutation is at least $\frac{1}{T_{init}+n}$ by the population size not larger than $T_{init} + n$ (i.e., Lemma 5), and the probability of applying deletion is $\frac{1}{3}$. Thus,

$$E_1 = l \cdot 3(T_{init} + n) = T_{init} \cdot 3(T_{init} + n),$$

where the 2nd '=' is by $l = T_{init}$ for the initial population.

In *phase 2*, let x^*_i denote the Pareto optimal solution with the first objective value W_i. We know that a subgraph with the minimum weight among all possible subgraphs with $k-1$ connected components can be generated by inserting the lightest edge which will not lead to a cycle into a subgraph with the minimum weight among all possible subgraphs with k connected components. By applying this property, x^*_{i+1} can be generated by inserting the lightest edge which will not lead to a cycle into x^*_i. Since the probability of selecting a specific solution for mutation is at least $\frac{1}{T_{init}+n}$ by Lemma 5, the probability of applying insertion is $\frac{1}{3}$, and the probability of selecting a specific edge from T for insertion is $\frac{1}{m}$, x^*_{i+1} can be generated in one step with probability of at least $\frac{1}{3m(T_{init}+n)}$ after x^*_i has been found. Note that x^*_0 has been found when this phase starts and a Pareto optimal solution will be never lost once it is found. Thus,

$$E_2 = (n-1)\cdot 3m(T_{init}+n).$$

By $\mathbb{E}[\![T_{init}]\!] = \frac{m}{m-1}$ and $\mathbb{E}[\![T^2_{init}]\!] \leqslant \frac{m}{m-3}$ in Lemma 1, the ERT of the whole process is at most

$$\mathbb{E}[\![E_1+E_2]\!] \in O(mn^2).$$

□

Thus, we have proved that SMO-GP solves the MST problem in $O(mn^2)$ ERT. Compared with the ERT $O(mn(n+\log w_{\max}))$ of GSEMO (a simple multi-objective GA) on MST reported in literature [27], SMO-GP has a better upper bound.

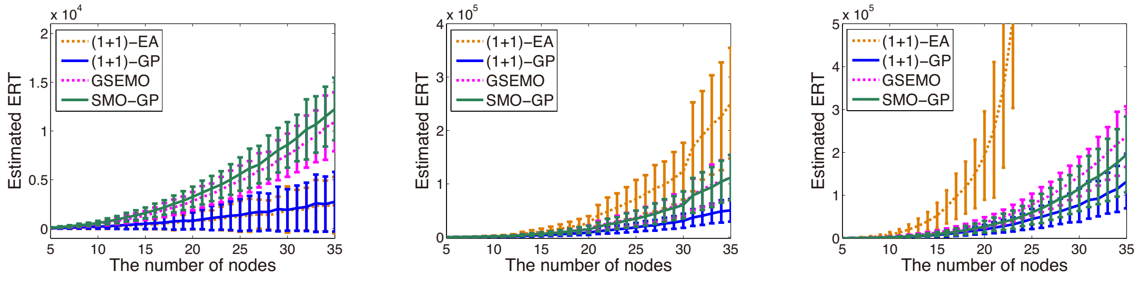
By comparing our analysis with the analysis of GSEMO in [27], we find that the better performance of SMO-GP is because the employed complexity objective leads to a more efficient process to find the empty graph in *phase 1* of the optimization. Thus, the variable solution structure of GP allows us to have a complexity measure for the solution, and by directly minimizing the complexity as an auxiliary objective, we have accelerated the optimization. This also discloses that if the solution complexity can be controlled, variable solution structure might be helpful for optimization.

4.4 Empirical studies

As the comparison between GPs and GAs on the MM problem, only the running time upper bounds on the MST problem make the comparison between GPs and GAs not quite strict. Therefore, we also conduct experiments to compare these algorithms on sparse, moderate, and dense graphs with the number of edges $m \in \Theta(n)$, $\Theta(n\sqrt{n})$, and $\Theta(n^2)$, respectively, which have been introduced to empirically investigate the performance of GPs and GAs on the MM problem.

For each kind of graph, the range of the number of nodes n is set to be $[5,35]$. On each size of n, we use the average running time of 1000 independent runs as an estimation of the ERT and also record the standard deviation. For each independent run, the graph is constructed by setting the weight of each edge be an integer randomly selected from $[1,n]$.

The experimental results are plotted in Figure 6. We can observe that the curves of GSEMO and SMO-GP are always very close. For (1+1)-EA and (1+1)-GP, their curves are nearly overlapped in Figure 6(a); and the curve of (1+1)-GP is much lower than that of (1+1)-EA in Figure 6(b–c). By applying $w_{\max} \leqslant n$ in our experimental setting to the derived running time bounds (i.e., the third column of Table 1), we can get Table 2, from which we can derive that GSEMO and SMO-GP behave the same for all graphs; (1+1)-GP performs as same as (1+1)-EA for sparse graph ($m \in \Theta(n)$), while (1+1)-GP is better for moderate ($m \in \Theta(n\sqrt{n})$) and dense ($m \in \Theta(n^2)$) graphs. Thus, the empirical results are consistent with the theoretical results.



(a) Sparse graph (b) Moderate graph (c) Dense graph

Figure 6 Estimated ERT comparison on the MST problem for different graphs.

Table 2 Comparison of running time on the MST problem when $w_{\max} \leqslant n$.

Problem	(1+1)-EA	(1+1)-GP	GSEMO	SMO-GP
MST	$O(m^2 \log n)$	$O(mn \log n)$	$O(mn^2)$	$O(mn^2)$

5 Conclusion

In this paper, we analyze the running time of GPs on two classical combinatorial problems, that is, the *MM* and the *MST* problems, for the first time. Both single-objective and multi-objective GPs are considered. The theoretical results show that GPs with tree-structured solutions achieve better performance than the previously analyzed GAs with binary-vectored solutions on these two problems, which is also verified by experiments. This result provides theoretical evidence to support the usefulness of rich representations in evolutionary optimization. From the analysis, we find that the variable solution structure might be helpful for evolutionary optimization when the solution complexity can be well controlled.

Acknowledgements
We want to thank the anonymous reviewers for helpful comments and suggestions. This research was supported by the National Science Foundation of China (61375061, 61333014) and the Jiangsu Science Foundation (BK2012303).

References

1 Bäck T. Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford: Oxford University Press, 1996
2 Goldberg D E. Genetic Algorithms in Search, Optimization, and Machine Learning. Reading: Addison-Wesley, 1989
3 Koza, J R. Genetic programming as a means for programming computers by natural selection. Stat Comput, 1994, 4: 87–112
4 Rothlauf F. Representations for Genetic and Evolutionary Algorithms. Berlin: Springer, 2006
5 Hoai N X, McKay R I, Essam D. Representation and structural difficulty in genetic programming. IEEE Trans Evol Comput, 2006, 10: 157–166
6 Poli R, Langdon W B, McPhee N F. A Field Guide to Genetic Programming. Barking: Lulu Enterprises, 2008
7 Koza J R. Human-competitive results produced by genetic programming. Genet Program Evol Mach, 2010, 11: 251–284
8 Khan S, Baig A R, Ali A, et al. Unordered rule discovery using Ant Colony Optimization. Sci China Inf Sci, 2014, 57: 1–15
9 Guo W, Liu G, Chen G, et al. A hybrid multi-objective PSO algorithm with local search strategy for VLSI partitioning. Front Comput Sci, 2014, 8: 203–216
10 Poli R, Vanneschi L, Langdon W B, et al. Theoretical results in genetic programming: The next ten years? Genet Program Evol Mach, 2010, 11: 285–320
11 Durrett G, Neumann F, O'Reilly U M. Computational complexity analysis of simple genetic programming on two problems modeling isolated program semantics. In: Proceedings of International Workshop on Foundations of Genetic Algorithms, Schwarzenberg, Austria, 2011. 69–80
12 Wagner M, Neumann F. Single- and multi-objective genetic programming: New runtime results for sorting. In: Proceedings of IEEE Congress on Evolutionary Computation, Beijing, China, 2014. 125–132
13 Kötzing T, Sutton A M, Neumann F, et al. The Max problem revisited: The importance of mutation in genetic programming. In: Proceedings of ACM Conference on Genetic and Evolutionary Computation, Philadelphia, PA, 2012. 1333–1340
14 Nguyen A, Urli T, Wagner M. Single- and multi-objective genetic programming: New bounds for weighted order and majority. In: Proceedings of International Workshop on Foundations of Genetic Algorithms, Adelaide, Australia, 2013. 161–172
15 Kötzing T, Neumann F, Spöhel R. PAC learning and genetic programming. In: Proceedings of ACM Conference on Genetic and Evolutionary Computation, Dublin, Ireland, 2011. 2091–2096
16 Neumann F. Computational complexity analysis of multi-objective genetic programming. In: Proceedings of ACM Conference on Genetic and Evolutionary Computation, Philadelphia, PA, 2012. 799–806
17 Wagner M, Neumann F. Parsimony pressure versus multi-objective optimization for variable length representations. In: Proceedings of International Conference on Parallel Problem Solving from Nature, Taormina, Italy, 2012. 133–142
18 He J, Yao X. Drift analysis and average time complexity of evolutionary algorithms. Artif Intell, 2001, 127: 57–85
19 Droste S, Jansen T, Wegener I. On the analysis of the (1+1) evolutionary algorithm. Theor Comput Sci, 2002, 276: 51–81

20 Auger A, Doerr B. Theory of Randomized Search Heuristics: Foundations and Recent Developments. Singapore: World Scientific, 2011

21 Neumann F, Witt C. Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity. Berlin: Springer-Verlag, 2010

22 Giel O, Wegener I. Evolutionary algorithms and the maximum matching problem. In: Proceedings of Annual Symposium on Theoretical Aspects of Computer Science, Berlin, Germany, 2003. 415–426

23 Giel O, Wegener I. Maximum cardinality matchings on trees by randomized local search. In: Proceedings of ACM Conference on Genetic and Evolutionary Computation, Seattle, WA, 2006. 539–546

24 Neumann F, Wegener I. Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. Theor Comput Sci, 2007, 378: 32–40

25 Doerr B, Johannsen D, Winzen C. Multiplicative drift analysis. Algorithmica, 2012, 64: 673–697

26 Raidl G R, Koller G, Julstrom B A. Biased mutation operators for subgraph-selection problems. IEEE Trans Evol Comput, 2006, 10: 145–156

27 Neumann F, Wegener I. Minimum spanning trees made easier via multi-objective optimization. Nat Comput, 2006, 5: 305–319

28 Yu Y, Zhou Z-H. A new approach to estimating the expected first hitting time of evolutionary algorithms. Artif Intell, 2008, 172: 1809–1832

29 Laumanns M, Thiele L, Zitzler E. Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions. IEEE Trans Evol Comput, 2004, 8: 170–182

30 Qian C, Yu Y, Zhou Z-H. An analysis on recombination in multi-objective evolutionary optimization. Artif Intell, 2013, 204: 99–119

31 Giel O, Wegener I. Evolutionary algorithms and the maximum matching problem. University of Dortmund Technical Report CI 142/02. 2002