Towards Theoretically Grounded Evolutionary Learning

Chao Qian*
State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210023, China
qianc@lamda.nju.edu.cn

Abstract

Machine learning tasks are often formulated as complex optimization problems, where the objective function can be non-differentiable, non-continuous, non-unique, inaccurate, dynamic, and have many local optima, making conventional optimization algorithms fail. Evolutionary Algorithms (EAs), inspired by Darwin's theory of evolution, are general-purpose randomized heuristic optimization algorithms, mimicking variational reproduction and natural selection. EAs have yielded encouraging outcomes for solving complex optimization problems (e.g., neural architecture search) in machine learning. However, due to the heuristic nature of EAs, most outcomes to date have been empirical and lack theoretical support, encumbering their acceptance to the general machine learning community. In this paper, I will review the progress towards theoretically grounded evolutionary learning, from the aspects of analysis methodology, theoretical perspectives and learning algorithms. Due to space limit, I will include a few representative examples and highlight our contributions. I will also discuss some future challenges.

1 Introduction

Machine learning tries to learn generalizable models from data, which can be generally divided into three components: model representation, evaluation and optimization [Domingos, 2012]. The way of model representation (e.g., deep neural networks) becomes more and more complex, and the function of model evaluation may not have good properties. Furthermore, the environment can be subject to a wide range of uncertainties. A machine learning task is thus often formulated as a complex optimization problem, whose objective function can be non-differentiable, non-continuous, non-unique, inaccurate, dynamic and have many local optima, requiring powerful optimization algorithms.

EAs [Bäck, 1996] are a type of general-purpose randomized heuristic optimization algorithms, by simulating the two key factors of natural evolution, variational reproduction and natural selection. Starting from an initial population of solutions, EAs iteratively reproduce offspring solutions by recombination and mutation, and select better ones from the parent and offspring solutions to form the next population. During the evolutionary process, EAs only require the solutions to be represented and their goodness to be evaluated, and thus can be applied in the "black-box" manner to solve optimization problems. The population-based search of EAs also matches the requirement of multi-objective optimization, i.e., EAs can generate a set of Pareto optimal solutions by running only once. Furthermore, natural evolution have been successfully processed in noisy and dynamic natural environments, and hence the algorithmic simulations are also likely to be able to handle noise and adapt to dynamic changes.

Due to the powerful optimization ability, EAs have been applied to solve complex optimization problems in machine learning, leading to the direction of *evolutionary learning* [Zhou *et al.*, 2019]. For example, Zhou *et al.* [2002] applied EAs to solve the ensemble pruning problem, generating a small subset of individual learners with strong generalization ability; Real *et al.* [2017] applied EAs to search the architecture of deep neural networks automatically, achieving competitive performance to the hand-designed models by experts; Wang *et al.* [2022] applied EAs to policy search for reinforcement learning in real-world scenarios, generating a set of policies with both high quality and diversity.

Though evolutionary learning has achieved successes, most outcomes to date (like those introduced above) have been empirical and lack theoretical support. In fact, theoretical analysis of EAs is quite difficult, due to their heuristic and randomized nature. The lack of sound theoretical foundation encumbers the acceptance of evolutionary learning by the general machine learning community. Next, I will review the progress towards theoretically grounded evolutionary learning, from the aspects of analysis methodology, theoretical perspectives, and algorithms with theoretical guarantees.

2 Analysis Methodology

Towards building the theoretical foundation of evolutionary learning, the first step is to develop general analysis tools that can guide the analysis of EAs on new problems, rather than to perform adhoc analysis starting from scratch. As EAs are used for optimization, running time complexity is one fundamental theoretical aspect, which characterizes how soon an

*This work was supported by the NSFC (62022039).

algorithm can solve a problem. Specifically, the running time complexity of an EA is often measured by the number of fitness (i.e., objective) evaluations required to find a desired solution for the first time. Much efforts thus have been put into developing general approaches for analyzing the running time complexity of EAs. In the following, I will introduce three representatives. Note that the process of optimizing a problem by an EA is often modeled as a Markov chain $\{\xi_t\}_{t=0}^{+\infty}$.

Fitness Level. Given a maximization problem f to be solved by an elitist EA which never loses the best found solution, the *fitness level* method [Wegener, 2000] first partitions the solution space into level sets $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_m$ according to the fitness value, i.e., $\forall i < j, \boldsymbol{s} \in \mathcal{S}_i, \boldsymbol{s}' \in \mathcal{S}_j : f(\boldsymbol{s}) < f(\boldsymbol{s}')$. Intuitively, the level sets form stairs; the higher, the better. By pessimistically assuming that a single jump can reach only the adjacent upper-level set, an upper bound on the expected running time of the EA can be derived by summing up the time taken for leaving every stair. Let $v_i \leq P(\xi_{t+1} \in \cup_{j=i+1}^{m} \mathcal{S}_j \mid \xi_t \in \mathcal{S}_i)$ denote a lower bound on the probability of jumping to higher levels. Then, the time taken for leaving stair $\mathcal{S}_i$ is at most $1/v_i$, leading to the upper bound $\sum_{j=i}^{m-1} 1/v_j$ on the expected running time starting from $\mathcal{S}_i$. By optimistically assuming that a single jump can reach the target level set, the expected running time of the EA can be lower bounded by the time of leaving a stair, which is at least $1/u_i$ where $u_i \geq P(\xi_{t+1} \in \cup_{j=i+1}^{m} \mathcal{S}_j \mid \xi_t \in \mathcal{S}_i)$ denotes an upper bound on the probability of jumping to higher levels. Based on this basic idea, advanced variants have also been proposed. For example, Sudholt [2013] considered the probability distribution that the EA jumps to higher levels carefully, leading to tighter lower bounds on the expected running time; Corus *et al.* [2017] proposed the level-based theorem for deriving upper bounds on the expected running time of non-elitist EAs.

Drift Analysis. By introducing a distance function $V(\cdot)$ to measure the distance from a state to the target state space, the *drift analysis* approach [He and Yao, 2001] estimates the average drift towards the target, i.e., $\mathbb{E}[V(\xi_t) - V(\xi_{t+1}) \mid \xi_t]$, of every step of an EA, and then derives an upper (lower) bound on the expected running time of the EA through dividing the initial distance by a lower (upper) bound on the one-step average drift. Many variants have been proposed, e.g., *multiplicative* drift analysis [Doerr *et al.*, 2012] is easier to use when the average drift is roughly proportional to the current distance $V(\xi_t)$ to the target space; *simplified negative* drift analysis [Oliveto and Witt, 2011] can be applied to prove exponential lower bounds on the running time when the average drift is a negative constant and the probability of jumping towards or away from the target space decays exponentially.

Switch Analysis. Different from the above two approaches which provide paths to be followed for deriving the expected running time of an EA solving an optimization problem from scratch, we developed the *switch analysis* approach [Yu *et al.*, 2015], by comparing the expected running time of the given EA process $\{\xi_t\}_{t=0}^{+\infty}$ with that of a reference EA process $\{\xi'_t\}_{t=0}^{+\infty}$, which can be specially designed to be somewhat similar to $\{\xi_t\}_{t=0}^{+\infty}$ but easier to be analyzed. The difference of these two chains at step k is calculated by the time difference of two intermediate chains $\{\xi_t^{k+1}\}_{t=0}^{+\infty}$ and $\{\xi_t^{k}\}_{t=0}^{+\infty}$, where ξ_t^k acts like the given chain ξ_t before time k, switches to the state space of the reference chain ξ'_t at time k, and then acts like ξ'_t. All these one-step differences are summed up to bound the total time difference of $\{\xi_t\}_{t=0}^{+\infty}$ and $\{\xi'_t\}_{t=0}^{+\infty}$. Thus, the analysis of the expected running time of a complex given process can be simplified by the comparison with a relatively simpler reference process. We have applied switch analysis to analyze the expected running time of EAs solving multi-objective optimization problems, and derived tighter bounds than before [Bian *et al.*, 2018]. For example, for the EA, namely SEMO, solving the m-objective problem mCOCZ where $m \geq 4$, the known bound $O(n^{m+1})$ [Laumanns *et al.*, 2004] is improved to be $O(n^m)$.

3 Theoretical Perspectives

Utilizing the general tools introduced in the previous section, a series of theoretical results have been attained, which can bring better understanding about the behaviors of EAs and offer some insight for algorithm design. In this section, I will focus on the common complex optimization problems, i.e., multi-objective, constrained, noisy and dynamic optimization problems, in machine learning, and introduce some representative results for EAs solving these problems.

Multi-objective Optimization. Many machine learning tasks involve multiple objectives. For example, ensemble pruning [Zhou *et al.*, 2002] tries to optimize the generalization performance of a selective ensemble using as few individual learners as possible; neural architecture search [Real *et al.*, 2017] hopes to find an architecture that maximizes accuracy and minimizes computation cost at the same time. Formally speaking, *multi-objective optimization* requires to simultaneously optimize two or more objective functions, i.e.,

$$\max_{\boldsymbol{s} \in \mathcal{S}} \ \big(f_1(\boldsymbol{s}), f_2(\boldsymbol{s}), ..., f_m(\boldsymbol{s})\big),$$

where $\mathcal{S}$ denotes the solution space, and $f_1, f_2, \ldots, f_m$ are m objective functions to be maximized. When there are two objective functions, it is also called *bi-objective optimization*.

As the objective functions are usually conflicting, it is impossible to have one solution which is optimal for all objectives. The comparison between solutions in multi-objective optimization is usually based on the *domination* relationship. That is, for two solutions $\boldsymbol{s}$ and $\boldsymbol{s}' \in \mathcal{S}$,

- $\boldsymbol{s}$ *weakly dominates* $\boldsymbol{s}'$, denoted as $\boldsymbol{s} \succeq \boldsymbol{s}'$, if $\forall i \in \{1, 2, \ldots, m\} : f_i(\boldsymbol{s}) \geq f_i(\boldsymbol{s}')$;
- $\boldsymbol{s}$ *dominates* $\boldsymbol{s}'$, denoted as $\boldsymbol{s} \succ \boldsymbol{s}'$, if $\boldsymbol{s} \succeq \boldsymbol{s}'$ and $\exists i \in \{1, 2, \ldots, m\} : f_i(\boldsymbol{s}) > f_i(\boldsymbol{s}')$.

If neither $\boldsymbol{s} \succeq \boldsymbol{s}'$ nor $\boldsymbol{s}' \succeq \boldsymbol{s}$, they are *incomparable*. A solution is *Pareto optimal* if there is no other solution in $\mathcal{S}$ that dominates it. The set of objective vectors of all the Pareto optimal solutions constitutes the *Pareto front*, which is just the goal of multi-objective optimization.

Due to the characteristic of population-based search, EAs have become a popular tool for multi-objective optimization. However, the theoretical analysis is still underdeveloped, especially compared to the single-objective scenario. We analyzed the influence of recombination operators, by comparing

the expected running time of the same EA with and without recombination for solving some bi-objective benchmark problems [Qian *et al.*, 2013]. The analysis discloses that the recombination operator can work by accelerating the filling of the Pareto front through recombining diverse Pareto optimal solutions found-so-far, which is unique to multi-objective optimization, as there is no Pareto front in single-objective situations. Recently, we analyzed the popular NSGA-II [Deb *et al.*, 2002], and proved that stochastic tournament selection (i.e., k tournament selection where k is uniformly sampled at random) can be better than the common binary tournament selection strategy, e.g., the expected running time for solving the LOTZ problem can be reduced from $O(n^3)$ to $O(n^2)$ by using stochastic tournament selection [Bian and Qian, 2022].

Constrained Optimization. The optimization problems in machine learning also often come with constraints. For example, to avoid overfitting, one often needs to minimize the error of a model, while constraining the model complexity. *Constrained optimization* can be generally formulated as

$$\max_{\boldsymbol{s}\in\mathcal{S}} \quad f(\boldsymbol{s}) \quad \text{s.t.} \quad \begin{array}{ll} g_i(\boldsymbol{s}) = 0 & \text{for } 1 \le i \le q, \\ h_i(\boldsymbol{s}) \le 0 & \text{for } q+1 \le i \le m, \end{array}$$

where f is the objective function, g_i and h_i are the equality and inequality constraints, respectively. A solution is (in)feasible if it does (not) satisfy the constraints. The goal is to find a feasible solution maximizing the objective f.

When EAs are applied to constrained optimization, a fundamental question is how to deal with the constraints. The most common way is the penalty function method, which transforms the original constrained optimization problem into an unconstrained one by adding the constraint violation degree to the objective f. However, some theoretical studies have shown that it can be better if the original constrained optimization problem is transformed into a bi-objective optimization problem that optimizes the original objective f and a constraint-related objective (e.g., the constraint violation degree) simultaneously. This way is called *Pareto optimization*. For example, Neumann and Wegener [2006] proved that for EAs solving the minimum spanning tree problem with dense graphs, using Pareto optimization can be faster than using penalty function, to find a minimum spanning tree; we proved that for EAs solving the NP-hard minimum cost coverage problem, using Pareto optimization can even be exponentially faster to achieve some approximation ratio [Qian *et al.*, 2015a]. By Pareto optimization, infeasible solutions can be naturally incorporated into the evolutionary process, which may bring a "shortcut" to the good feasible solutions.

Noisy Optimization. In machine learning, the objective evaluation of solutions can be inaccurate, resulting in *noisy optimization*. For example, a prediction model is usually evaluated on a limited amount of data, where the estimated performance generally has some deviation from the true one. Let f and F denote the true and noisy objective functions, respectively. A common noise model is the multiplicative one, i.e., $F(\boldsymbol{s}) = f(\boldsymbol{s}) \cdot \delta$. The occurrence of noise may mislead the search direction of EAs, making the optimization inefficient. We proved that the robustness of EAs against noise can be increased by the strategies of sampling [Qian *et al.*, 2018a], threshold selection [Qian *et al.*, 2018b] and using large population [Qian *et al.*, 2021]. For example, for the (1+1)-EA solving the OneMax problem under high noise levels, using threshold selection can reduce the expected running time of finding the optimal solution from exponential to polynomial [Qian *et al.*, 2018b]. By threshold selection, EAs accept an offspring solution only if its objective value is better than that of a parent solution by at least a threshold, reducing the risk of accepting a bad solution or deleting a good solution.

Dynamic Optimization. The real-world environment where machine learning tasks are performed often changes dynamically, resulting in *dynamic optimization*. That is, the objective, constraint or solution space of the optimization problem may change over time. EAs are believed to be able to adapt to the dynamic changes quickly, and have been widely applied to dynamic optimization in practice. However, theoretical analysis is often lacked. Neumann and Witt [2015] gave the theoretical evidence by proving that for the dynamic makespan scheduling problem, even the simple EA, (1+1)-EA, can maintain a good discrepancy efficiently when the processing time of one job changes dynamically. By considering EAs on linear functions under dynamic uniform constraints, population has also been proved to be necessary in tracking changes dynamically [Shi *et al.*, 2019].

4 Theoretically Grounded Evolutionary Learning Algorithms

Inspired by the theoretical results as those introduced in the previous section, a series of evolutionary learning algorithms with bounded approximation guarantees have been developed. In this section, I will introduce those algorithms for three representative learning problems, i.e., ensemble pruning, subset selection and result diversification.

Ensemble Pruning. Given a set of trained individual learners, *ensemble pruning* tries to select a small subset of individual learners to comprise the ensemble, instead of combining all. There are naturally two goals, i.e., maximizing the generalization performance and minimizing the number of selected individual learners. Previous methods mainly optimize the single objective, i.e, the generalization performance, by using greedy algorithms or EAs. We proposed an EA to solve the explicit bi-objective formulation, achieving better performance than state-of-the-art ensemble pruning methods both theoretically and empirically [Qian *et al.*, 2015b].

Subset Selection. It is a more general problem, aiming to select a limited number of items from $V = \{v_1, v_2, \ldots, v_n\}$ for optimizing some given objective $f : 2^V \to \mathbb{R}$. That is,

$$\max_{\boldsymbol{s}\in\{0,1\}^n} f(\boldsymbol{s}) \quad \text{s.t.} \quad |\boldsymbol{s}| \le k, \tag{1}$$

where k is a given budget, and $\boldsymbol{s} \in \{0,1\}^n$ represents a subset of V where $s_i = 1$ if v_i is selected and $s_i = 0$ otherwise. It is NP-hard in general, and has many applications, such as sparse regression, influence maximization, document summarization and sensor placement, just to name a few.

Inspired by the theoretical results of applying Pareto optimization to handle constraints, we proposed the POSS algorithm [Qian *et al.*, 2015c], which reformulates the original

Algorithm 1 POSS Algorithm

1: Let $P \leftarrow \{\mathbf{0}\}$;
2: **while** some criterion is not met
3: Choose $\boldsymbol{s}$ from P uniformly at random;
4: Create $\boldsymbol{s}'$ by flipping each bit of $\boldsymbol{s}$ with prob. $1/n$;
5: **if** $\nexists \boldsymbol{z} \in P$ such that $\boldsymbol{z} \succ \boldsymbol{s}'$ **then**
6: $P \leftarrow (P \setminus \{\boldsymbol{z} \in P \mid \boldsymbol{s}' \succeq \boldsymbol{z}\}) \cup \{\boldsymbol{s}'\}$
7: **return** $\arg\max_{\boldsymbol{s} \in P, |\boldsymbol{s}| \leq k} f(\boldsymbol{s})$

constrained problem in Eq. (1) into a bi-objective problem

$$\max_{\boldsymbol{s} \in \{0,1\}^n} \quad (f(\boldsymbol{s}), -|\boldsymbol{s}|), \tag{2}$$

i.e., maximizing the original objective $f(\boldsymbol{s})$ and minimizing the subset size $|\boldsymbol{s}| = \sum_{i=1}^{n} s_i$ simultaneously. POSS employs a simple EA [Laumanns *et al.*, 2004] to solve this bi-objective problem, as shown in Algorithm 1. Starting from the all-0s vector $\mathbf{0}$ (i.e., the empty set), POSS iteratively improves the quality of solutions in the population P. In each iteration, a parent solution $\boldsymbol{s}$ is selected from P uniformly at random (line 3), and used to generate an offspring solution $\boldsymbol{s}'$ by bitwise mutation (line 4), which flips each bit of $\boldsymbol{s}$ independently with probability $1/n$. The newly generated $\boldsymbol{s}'$ is then used to update P (lines 5–6). If $\boldsymbol{s}'$ is not dominated by any solution in P (line 5), it will be added into P, and those parent solutions weakly dominated by $\boldsymbol{s}'$ will be deleted (line 6). When POSS terminates, the best feasible solution w.r.t. the original problem (i.e., $\arg\max_{\boldsymbol{s} \in P, |\boldsymbol{s}| \leq k} f(\boldsymbol{s})$) will be selected from the final population P as the output (line 7).

POSS can achieve the best-known polynomial-time approximation guarantees for subset selection. When the objective f is monotone submodular, it achieves the approximation ratio of $1 - 1/e$ [Friedrich and Neumann, 2015], which is optimal [Nemhauser and Wolsey, 1978]. When f is monotone approximately submodular, it achieves the approximation $1 - e^{-\gamma}$ [Qian *et al.*, 2015c], which is optimal as well [Harshaw *et al.*, 2019]. Note that γ is the submodularity ratio characterizing the closeness of f to submodularity. Even when f is non-monotone, POSS can achieve the approximation guarantee, $f(\boldsymbol{s}) \geq (1 - 1/e) \cdot (\mathrm{OPT} - k\epsilon)$ [Qian *et al.*, 2019], where OPT denotes the optimal function value and $\epsilon \geq 0$ captures the degree of approximate monotonicity of f. This also reaches the best-known one [Krause *et al.*, 2008].

Noisy Subset Selection. For subset selection, there are many practical situations where the objective evaluation can be noisy. For example, in sparse regression, only a set of limited data can be used for evaluation, bringing noise. Inspired by the theoretical results of applying threshold selection to deal with noise, we incorporated this strategy into POSS, resulting in the PONSS algorithm [Qian *et al.*, 2017]. Consider the multiplicative noise with $\delta \in [1 - \epsilon, 1 + \epsilon]$, i.e., $(1 - \epsilon) \cdot f(\boldsymbol{s}) \leq F(\boldsymbol{s}) \leq (1 + \epsilon) \cdot f(\boldsymbol{s})$. A solution $\boldsymbol{s}$ weakly dominating $\boldsymbol{s}'$ in PONSS requires

$$F(\boldsymbol{s}) \geq ((1 + \epsilon)/(1 - \epsilon)) \cdot F(\boldsymbol{s}') \ \wedge \ |\boldsymbol{s}| \leq |\boldsymbol{s}'|,$$

instead of the previous $F(\boldsymbol{s}) \geq F(\boldsymbol{s}') \wedge |\boldsymbol{s}| \leq |\boldsymbol{s}'|$ in POSS. By this conservative comparison, solutions with similar F values will be kept in the population P, reducing the risk of removing a good solution. We proved that PONSS achieves an approximation ratio of $\frac{1-\epsilon}{1+\epsilon}(1 - e^{-\gamma})$, significantly better than that of the greedy algorithm [Horel and Singer, 2016].

Dynamic Subset Selection. In real-world applications of subset selection, the budget k on the size constraint $|\boldsymbol{s}| \leq k$ may change over time, reflecting the change of resources. It has been proved that whether k decreases or increases, POSS can always maintain a good approximation ratio quickly [Bian *et al.*, 2021; Roostapour *et al.*, 2022]. In fact, this result even holds for the more general cost constraint $c(\boldsymbol{s}) \leq k$, where $c(\cdot)$ is a cost function.

Large-scale Subset Selection. Though POSS can achieve good approximation guarantees in general as well as excellent empirical performance in various applications of subset selection, its running time may be unsatisfactory for large-scale situations. For acceleration, Crawford [2021] introduced a biased selection strategy to select solutions for mutation, while we developed a parallel version of POSS, which generates as many offspring solutions as the number of processors in parallel in each iteration [Qian *et al.*, 2016], as well as a distributed version, which employs a divide-and-conquer strategy to partition the data set into multiple machines and run POSS on each machine in parallel [Qian, 2020]. All these accelerated variants still enjoy good approximation guarantees, though with a little loss.

Result Diversification. Some applications (e.g., feature selection, document summarization and web-based search) may require the subset to have not only high "quality" but also high "diversity", leading to the result diversification problem

$$\max_{\boldsymbol{s} \in \{0,1\}^n} f(\boldsymbol{s}) + \lambda \cdot div(\boldsymbol{s}) \quad \text{s.t.} \quad |\boldsymbol{s}| \leq k,$$

where f and div represent the quality and diversity, respectively. For some common diversity measures, we proved that POSS with carefully-designed bi-objective reformulation can still achieve good polynomial-time approximation guarantees [Qian *et al.*, 2022]. Even for the more general matroid constraint, POSS can achieve an asymptotically optimal approximation ratio, $1/2 - \epsilon/(4n)$, where $\epsilon > 0$. Furthermore, when f or div changes dynamically, POSS can maintain this approximation ratio in polynomial running time, addressing the open question [Borodin *et al.*, 2017].

5 Discussion

Though there have been progresses towards theoretically grounded evolutionary learning as introduced before, many works remain to be done. For example, we still need general running time analysis tools which are not only powerful but also easy to use. It would be interesting to consider EAs for robust optimization, which also appears frequently in machine learning. Of course, more theoretically grounded evolutionary learning algorithms are expected, and we are very looking forward to those algorithms that can achieve an approximation guarantee better than the best-known one.

Besides, how to make evolutionary learning efficient is another major challenge. There are many possible ways, e.g., combining EAs with gradient-based optimization methods, self-adjusting the parameters of EAs by learning, or reproducing offspring solutions with the help of surrogate models.

References

[Bäck, 1996] T. Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, 1996.

[Bian and Qian, 2022] C. Bian and C. Qian. Better running time of the non-dominated sorting genetic algorithm ii (NSGA-II) by using stochastic tournament selection. In *PPSN*, 2022.

[Bian *et al.*, 2018] C. Bian, C. Qian, and K. Tang. A general approach to running time analysis of multi-objective evolutionary algorithms. In *IJCAI*, 2018.

[Bian *et al.*, 2021] C. Bian, C. Qian, F. Neumann, and Y. Yu. Fast Pareto optimization for subset selection with dynamic cost constraints. In *IJCAI*, 2021.

[Borodin *et al.*, 2017] A. Borodin, A. Jain, H. C. Lee, and Y. Ye. Max-sum diversification, monotone submodular functions, and dynamic updates. *ACM TALG*, 13(3):1–25, 2017.

[Corus *et al.*, 2017] D. Corus, D.-C. Dang, A. Eremeev, and P. K. Lehre. Level-based analysis of genetic algorithms and other search processes. *IEEE TEvC*, 22(5):707–719, 2017.

[Crawford, 2021] V. G. Crawford. Faster guarantees of evolutionary algorithms for maximization of monotone submodular functions. In *IJCAI*, 2021.

[Deb *et al.*, 2002] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE TEvC*, 6(2):182–197, 2002.

[Doerr *et al.*, 2012] B. Doerr, D. Johannsen, and C. Winzen. Multiplicative drift analysis. *Algorithmica*, 64(4):673–697, 2012.

[Domingos, 2012] P. Domingos. A few useful things to know about machine learning. *CACM*, 55(10):78–87, 2012.

[Friedrich and Neumann, 2015] T. Friedrich and F. Neumann. Maximizing submodular functions under matroid constraints by evolutionary algorithms. *ECJ*, 23(4):543–558, 2015.

[Harshaw *et al.*, 2019] C. Harshaw, M. Feldman, J. Ward, and A. Karbasi. Submodular maximization beyond non-negativity: Guarantees, fast algorithms, and applications. In *ICML*, 2019.

[He and Yao, 2001] J. He and X. Yao. Drift analysis and average time complexity of evolutionary algorithms. *AIJ*, 127(1):57–85, 2001.

[Horel and Singer, 2016] T. Horel and Y. Singer. Maximization of approximately submodular functions. In *NIPS*, 2016.

[Krause *et al.*, 2008] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *JMLR*, 9:235–284, 2008.

[Laumanns *et al.*, 2004] M. Laumanns, L. Thiele, and E. Zitzler. Running time analysis of multiobjective evolutionary algorithms on pseudo-Boolean functions. *IEEE TEvC*, 8(2):170–182, 2004.

[Nemhauser and Wolsey, 1978] G. L. Nemhauser and L. A. Wolsey. Best algorithms for approximating the maximum of a submodular set function. *MOR*, 3(3):177–188, 1978.

[Neumann and Wegener, 2006] F. Neumann and I. Wegener. Minimum spanning trees made easier via multi-objective optimization. *Natural Computing*, 5(3):305–319, 2006.

[Neumann and Witt, 2015] F. Neumann and C. Witt. On the runtime of randomized local search and simple evolutionary algorithms for dynamic makespan scheduling. In *IJCAI*, 2015.

[Oliveto and Witt, 2011] P. S. Oliveto and C. Witt. Simplified drift analysis for proving lower bounds in evolutionary computation. *Algorithmica*, 59(3):369–386, 2011.

[Qian *et al.*, 2013] C. Qian, Y. Yu, and Z.-H. Zhou. An analysis on recombination in multi-objective evolutionary optimization. *AIJ*, 204:99–119, 2013.

[Qian *et al.*, 2015a] C. Qian, Y. Yu, and Z.-H. Zhou. On constrained Boolean Pareto optimization. In *IJCAI*, 2015.

[Qian *et al.*, 2015b] C. Qian, Y. Yu, and Z.-H. Zhou. Pareto ensemble pruning. In *AAAI*, 2015.

[Qian *et al.*, 2015c] C. Qian, Y. Yu, and Z.-H. Zhou. Subset selection by Pareto optimization. In *NIPS*, 2015.

[Qian *et al.*, 2016] C. Qian, J.-C. Shi, Y. Yu, K. Tang, and Z.-H. Zhou. Parallel Pareto optimization for subset selection. In *IJCAI*, 2016.

[Qian *et al.*, 2017] C. Qian, J.-C. Shi, Y. Yu, K. Tang, and Z.-H. Zhou. Subset selection under noise. In *NIPS*, 2017.

[Qian *et al.*, 2018a] C. Qian, Y. Yu, K. Tang, Y. Jin, X. Yao, and Z.-H. Zhou. On the effectiveness of sampling for evolutionary optimization in noisy environments. *ECJ*, 26(2):237–267, 2018.

[Qian *et al.*, 2018b] C. Qian, Y. Yu, and Z.-H. Zhou. Analyzing evolutionary optimization in noisy environments. *ECJ*, 26(1):1–41, 2018.

[Qian *et al.*, 2019] C. Qian, Y. Yu, K. Tang, X. Yao, and Z.-H. Zhou. Maximizing submodular or monotone approximately submodular functions by multi-objective evolutionary algorithms. *AIJ*, 275:279–294, 2019.

[Qian *et al.*, 2021] C. Qian, C. Bian, Y. Yu, K. Tang, and X. Yao. Analysis of noisy evolutionary optimization when sampling fails. *Algorithmica*, 83(4):940–975, 2021.

[Qian *et al.*, 2022] C. Qian, D.-X. Liu, and Z.-H. Zhou. Result diversification by multi-objective evolutionary algorithms with theoretical guarantees. *AIJ*, 309:103737, 2022.

[Qian, 2020] C. Qian. Distributed Pareto optimization for large-scale noisy subset selection. *IEEE TEvC*, 24(4):694–707, 2020.

[Real *et al.*, 2017] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. Le, and A. Kurakin. Large-scale evolution of image classifiers. In *ICML*, 2017.

[Roostapour *et al.*, 2022] V. Roostapour, A. Neumann, F. Neumann, and T. Friedrich. Pareto optimization for subset selection with dynamic cost constraints. *AIJ*, 302:103597, 2022.

[Shi *et al.*, 2019] F. Shi, M. Schirneck, T. Friedrich, T. Kötzing, and F. Neumann. Reoptimization time analysis of evolutionary algorithms on linear functions under dynamic uniform constraints. *Algorithmica*, 81:828–857, 2019.

[Sudholt, 2013] D. Sudholt. A new method for lower bounds on the running time of evolutionary algorithms. *IEEE TEvC*, 17(3):418–435, 2013.

[Wang *et al.*, 2022] Y. Wang, K. Xue, and C. Qian. Evolutionary diversity optimization with clustering-based selection for reinforcement learning. In *ICLR*, 2022.

[Wegener, 2000] I. Wegener. On the expected runtime and the success probability of evolutionary algorithms. In *WG*, 2000.

[Yu *et al.*, 2015] Y. Yu, C. Qian, and Z.-H. Zhou. Switch analysis for running time analysis of evolutionary algorithms. *IEEE TEvC*, 19(6):777–792, 2015.

[Zhou *et al.*, 2002] Z.-H. Zhou, J. Wu, and W. Tang. Ensembling neural networks: Many could be better than all. *AIJ*, 137(1-2):239–263, 2002.

[Zhou *et al.*, 2019] Z.-H. Zhou, Y. Yu, and C. Qian. *Evolutionary Learning: Advances in Theories and Algorithms*. Springer, 2019.