Chapter 16
Stochastic Search in Metaheuristics

Walter J. Gutjahr and Roberto Montemanni

Abstract Stochastic search is a key mechanism underlying many metaheuristics. The chapter starts with the presentation of a general framework algorithm in the form of a stochastic search process that contains a large variety of familiar metaheuristic techniques as special cases. Based on this unified view, questions concerning convergence and runtime are discussed at the level of a theoretical analysis. Concrete examples from diverse metaheuristic fields are given. In connection with runtime results, important topics such as instance difficulty, phase transitions, parameter choice, No-Free-Lunch theorems or fitness landscape analysis are addressed. Furthermore, a short sketch of the theory of black-box optimization is given, and generalizations of results to stochastic search under noise and to robust optimization are outlined.

16.1 Introduction

The aim of this chapter is to present a unified view of *stochastic search* which is used as a core mechanism in many metaheuristics. Not every metaheuristic applies a probabilistic mechanism to organize the exploration of the search space; there are,

W. J. Gutjahr (✉)
University of Vienna, Vienna, Austria
e-mail: walter.gutjahr@univie.ac.at

R. Montemanni
Dalle Molle Institute for Artificial Intelligence, University of Applied Sciences of Southern Switzerland, Manno, Switzerland
e-mail: roberto.montemanni@supsi.ch

M. Gendreau, J.-Y. Potvin (eds.), *Handbook of Metaheuristics*,
International Series in Operations Research & Management Science 272,
https://doi.org/10.1007/978-3-319-91086-4_16

e.g., deterministic versions of Tabu Search. Interestingly enough, however, the incorporation of "random" (or more precisely: pseudo-random) steps into the algorithmic design is rather the usual than the exceptional case in the field of metaheuristics. Thus, it makes sense to have a closer look at this feature.

One would expect that all metaheuristics that perform stochastic search have some properties in common. Admittedly, at the moment, we are still far away from a general theory containing every stochastic metaheuristic as a special case. Nevertheless, some observations are available that are not restricted to a particular metaheuristic algorithm, but have been made, possibly in different appearance, for several seemingly unrelated algorithms.

The emphasis of this chapter is on results that lead to a deeper understanding of principles and properties common to more than one stochastic metaheuristic. Because of this goal, we concentrate on *theoretical* results, which can be rigorous or (at least) precise, where "rigorous" is understood in a mathematical sense, and "precise" means that some form of analytic derivation (although not necessarily a rigorous one) is used for predicting numerical experimental outcomes. It is clear that experimental results are at least as important—presumably even more important. However, they usually contribute to a smaller degree to a unifying understanding, so we shall not focus on them here.

The chapter is organized as follows: In Sect. 16.2, we develop a common formal framework capturing the essential features of most stochastic metaheuristics, and we shortly address the motivation for applying *stochastic* search in metaheuristic algorithms. Sections 16.3 and 16.4 are devoted to convergence results and to results dealing with required optimization time, respectively. The practically important issue of parameter choice in metaheuristics is briefly outlined in Sect. 16.5. Section 16.6 discusses "No-Free-Lunch" theorems and their implications for stochastic search, in particular the desirability of a problem-specific fitness landscape analysis. Some techniques for the latter are outlined in Sect. 16.7. The purest form of stochastic search algorithms are (stochastic) black-box optimizers, which are discussed in Sect. 16.8. Section 16.9 outlines an important special application area of metaheuristics, namely optimization under uncertainty or noise. Section 16.10 addresses robust optimization approaches, and Sect. 16.11 concludes the chapter.

16.2 General Framework

The aim of the stochastic search algorithms investigated in this chapter is the exact or approximative solution of combinatorial optimization (CO) problems of the form

$$\min f(x) \text{ such that } x \in S, \tag{16.1}$$

where S is a finite search space, f is a real-valued function called *objective function*, and "min" can be replaced by "max". The function f is also called *cost function* (if to be minimized) or *fitness function* (if to be maximized). We consider iterative

algorithms A of the following general type: In iteration t, algorithm A uses a memory M_t and a list L_t of solutions $x_i \in S$. The list L_t contains new "trial points" for the optimization. The algorithm proceeds as follows:

1. Initialize M_1 according to some rule.
2. In iteration $t = 1, 2, \ldots$, until some stopping criterion is satisfied,
   a. determine the list L_t as a function $g(M_t, z_t)$ of M_t and of a random influence z_t;
   b. determine the objective function values $f(x_i)$ of all $x_i \in L_t$, and form a list L_t^+ containing the pairs $(x_i, f(x_i))$;
   c. determine the new memory content M_{t+1} as a function $h(M_t, L_t^+, z_t')$ of the current M_t, of the list of solution-value pairs L_t^+, and of a random influence z_t'.

The currently proposed (approximate) solution x_t^{curr} in iteration t results as some function of (M_t, L_t^+) specified by A. Also the stopping criterion defined by A depends on (M_t, L_t^+).

In this formalism, one can imagine z_t and z_t' as vectors of (pseudo-)random numbers that are used by the stochastic algorithm. The function $g(M_t, z_t)$ specifies, for a given memory M_t, a probability distribution for the list of new search points; the function $h(M_t, L_t^+, z_t')$ specifies, given memory M_t and current list L_t^+ of solution-value pairs, a probability distribution for the new content of the memory. If the functions g and h are independent of z_t resp. z_t', we obtain the special case of a *deterministic* search algorithm.

The generic algorithm above, which is an extension of the generic black-box optimizer presented in [21] (discussed in Sect. 16.8), covers most—if not all—stochastic metaheuristics. We shall outline this by giving two special examples:

- *Simulated Annealing (SA)*: A neighborhood structure on S is used. M_t consists of a single element, the current search point x. Also L_t consists of a single element, the currently investigated neighbor solution y to x. To determine L_t from M_t, choose a random neighbor y to the element x in M_t. To update M_t to M_{t+1}, decide by the stochastic acceptance rule used in SA whether y is accepted or not. If yes, M_{t+1} contains y, otherwise it contains x.
- *Canonical Genetic Algorithm (GA)*: M_t consists of k solutions, and L_t also consists of k solutions. To determine L_t from M_t, apply the operators *mutation* and *crossover* to the solutions in M_t. This yields L_t. To update M_t to M_{t+1}, apply fitness-proportional *selection* to the population contained in L_t, using the corresponding objective function values. The result gives M_{t+1}.

In principle, the functions g and h may use any information on the problem instance. The important special case where g and h are only allowed to use the knowledge of the search space S and of the problem type, but not of the specific problem instance, is denoted as *black-box optimization* and will be dealt with in Sect. 16.8.

An important observation is that by construction, the "states" (M_t, L_t^+) visited during the execution of the algorithm form a *Markov process* in discrete time[1]: The distribution of the next state (M_{t+1}, L_{t+1}^+) only depends on the current state (M_t, L_t^+). Considering the objective function f as given, (M_t) $(t = 1, 2, \ldots)$ can already be seen as a Markov process, since the distribution of M_{t+1} only depends on M_t (via L_t^+, which results from M_t). This allows the application of Markov process theory to the analysis of stochastic search algorithms.

We may use the described algorithmic framework for giving a rough classification of several stochastic metaheuristics:

1. *Stochastic Local Search Algorithms:* Examples are Iterated Local Search (ILS), Simulated Annealing (SA), Generalized Hillclimbers (GHCs), or Variable Neighborhood Search (VNS). M_t contains a small, fixed number of solutions (e.g., incumbent solution, current search point, current neighbor) derived by using a neighborhood structure on S.
2. *Population-Based Stochastic Search Algorithms:* Examples are GAs and basic forms of Estimation-of-Distribution Algorithms (EDAs). M_t contains a "population" of solutions. The size of this population is a parameter of the algorithm.
3. *Model-Based Stochastic Search Algorithms:* The concept of model-based search has been introduced by Zlochin et al. [114]. This group of metaheuristics contains Ant Colony Optimization (ACO), some more elaborated forms of EDAs, or Cross-Entropy Optimization. Here, M_t consists of a vector of real-valued parameters, e.g., a pheromone vector in ACO, sometimes also of additional information.

Although metaheuristics as *Particle Swarm Optimization* (PSO) or some variants of *Evolution Strategies* (ES) do not deal with CO problems, but rather with *continuous* search spaces S instead, these metaheuristics can be used for CO problems as well by means of suitable problem encodings. For example, the Binary PSO algorithm proposed by Kennedy and Eberhart [67] maintains vectors interpreted as positions, best positions and velocities of "particles", from which discrete solutions can be derived by a probabilistic mechanism. In the classification above, this leads us to the model-based class with M_t containing a list of vectors of real numbers.

We close this section with the question of the general motivation for introducing *stochastic* elements (the random variables z_t and z_t' above) into a metaheuristic. Perhaps the simplest reason is that care must be taken to prevent a search algorithm from cycling through a small portion of the search space. Let us look at a simple example. Suppose we perform a search in the set $S = \{0,1\}^n$ of binary strings of length n, with some cost function f on S. For simplicity, let us suppose that all occurring cost values are different from each other. Our algorithm always stores the current solution x as well as the solution w visited just before x, and it iteratively moves from the current x to the lowest-cost neighbor solution y of x different from w, where x and y are called neighbors if they differ exactly in one bit position. This deterministic search algorithm is able to quickly find a locally optimal solution (i.e.,

[1] Since g and h do not depend on the iteration counter t, the Markov process is homogeneous. Dependence on t can easily be modeled by adding t as a component to the memory M_t.

a solution that does not have a neighbor with lower cost), but neither is it able to stop at a local optimum x_{loc}, nor does it typically leave the neighborhood of some suboptimal x_{loc} in order to continue the search for the global optimum. Of course, this undesirable behavior can be avoided by increasing our "tabu list" (consisting only of w in the naive algorithm above), but this comes at the price of increased computational cost. An alternative way to give the search process the freedom to leave local optima is to allow *random* moves from a solution point to a neighbor. Whenever we choose this alternative, it is much easier to ensure that no point in the solution space is excluded from the search in advance.

16.3 Convergence Results

The search process (M_t, L_t^+) is only helpful if it leads us to an optimal solution of (16.1) or, at least, to a good approximation. Ideally, the current solution x_t^{curr} derived from the state at time t becomes an element of the set S^* of optimal solutions in some iteration t_1 and remains unchanged in subsequent iterations. This behavior is denoted as convergence to the optimum. Since we consider *stochastic* search algorithms, the definition of convergence has to be modified. In probability theory, there are several different notions for the convergence of a stochastic process. One of the most natural in our context is *convergence in probability*: A stochastic search algorithm A converges to the optimum in probability, if the probability of the event $x_t^{curr} \in S^*$ converges (in the mathematical sense of the word) to unity as $t \to \infty$.

Convergence to the optimum in probability can be achieved easily even by simple stochastic search algorithms: Consider the (usually very inefficient) *random search* algorithm, where, in each iteration, L_t consists of a single solution x_t that is chosen at random from S according to some fixed distribution *independently* of M_t (and hence of the previous iterations). Let M_t contain the *best-so-far* solution x_t^{bsf} encountered up to iteration $t-1$: The variable x_t^{bsf} is initialized arbitrarily for $t = 1$ and is set to x_t in each iteration where $f(x_t)$ turns out to be better than $f(x_t^{bsf})$. If for each iteration t, we choose the currently proposed solution x_t^{curr} as the best-so-far solution x_t^{bsf}, random search converges to the optimum in probability. However, the runtime until hitting an optimal solution may be huge.

Interestingly, some more efficient algorithms (from a practical point of view) do *not* share the mentioned convergence property. For example, Rudolph [88] showed that the canonical GA, as described in the previous section, with x_t^{curr} defined as the best element of the current generation, *never* converges to the optimum in probability; this is simply due to the fact that by possible mutations, the probability of the event that the current population does not contain an element from S^* has always a strictly positive lower bound. By adding the "elite" solution x_t^{bsf} as an additional

component to the memory M_t, the algorithm can be made convergent to the optimum in probability.[2]

For a stochastic search algorithm A, it would be desirable that not only the probability of $x_t^{curr} \in S^*$ converges to one, but that exploitation of the search history increases the average fitness of the sample points, i.e., of the elements of L_t, which is not the case for random search. If for an algorithm A, the part of the memory M_t responsible for the generation of the list L_t of sample points converges to some state supporting only optimal (or at least good) solutions, one can expect that the quality of the sample points will improve during the process. Thus, the search algorithm will arrive at the optimum faster than random search.

Convergence results of the last kind are harder to show (and require stricter conditions on algorithms and parameter choices), but there exist such results in the literature for several metaheuristics. The first ones were derived for SA. In the case of SA with a logarithmic cooling scheme, Hajek [51] gave necessary and sufficient conditions for the current search point x_t (the solution contained in M_t) to converge in probability to S^*. Contrary to the best-so-far solution x_t^{bsf} which does not influence the process itself, the current search point x_t defines the next sample point candidates and thus determines the distribution of L_t. If x_t gradually focuses more and more on promising regions of the search space instead of doing "blind" random search (as in the first, high-temperature phase of SA), the chance of detecting the global optimum is increased compared to the random search algorithm. Therefore, convergence of x_t is more meaningful than convergence of x_t^{bsf} only.

In the ACO case, the "sample-generating" part of the memory M_t consists of the vector τ_t of pheromone values that determine the distribution of the solutions to be sampled in the current iteration. For algorithms of the MAX-MIN-Ant-System type developed by Stützle and Hoos [95] using "elitism" (i.e., incorporating also x_t^{bsf} into M_t), conditions are given in [40, 92] to ensure not only that x_t^{bsf} converges to the optimum, but also that τ_t converges to a limiting vector that only allows the generation of an optimal solution. A related result for Cross-Entropy Optimization was shown by Margolin [70].

What have these results for different metaheuristics in common? Typically, when proving a "strong" form of convergence for a stochastic search algorithm in the just-mentioned sense, the parametrization of the algorithm has to be chosen in such a way that a proper balance between *exploration* and *exploitation* is preserved: When the emphasis is too much on the exploration pole, random-search-type behavior results, and the sample-generating part of the memory M_t does not converge at all. On the other hand, when exploitation is emphasized too much, one does obtain convergence, but it is usually "premature" convergence to a suboptimal solution. By keeping the balance, convergence is still ensured, but slowed down to allow the detection of a global optimum. The specific form of the exploration-exploitation tradeoff depends on the algorithm under consideration. For example, for SA, high values of the temperature parameter favor exploration, low values favor exploitation.

[2] Elitism as a mechanism ensuring convergence of a GA has already been analyzed in [52], which appears to be the first paper on GA convergence.

In some stochastic metaheuristics, the question of convergence is conveniently addressed via a *system dynamics* approach. For example, Trelea [101] identifies *attractors*, i.e., stable fixed points of a dynamic process concretizing our generic (M_t, L_t^+) dynamics in the context of PSO. In the case of convergence, only attractors can be limiting points. In [101], the exploration-exploitation tradeoff and its connection to parameter choice is also explicitly addressed.

A very small selection of convergence results for stochastic metaheuristics have been mentioned in this section. For some other results, see, e.g., [64, 102] (GHCs), [37, 38] (EDAs), [39, 96] (ACO), or [50] (VNS).

16.4 Runtime Results

From the viewpoint of applications, the question of whether and in which sense a stochastic search algorithm A converges is less relevant than the question of what amount of computation times A requires for finding an optimal or a sufficiently good solution. Nevertheless, theoretical investigations must start with the convergence issue, since important performance measures are undefined or infinite if A has a nonzero probability of *never* arriving at an optimum, as, e.g., in the case of premature convergence.[3]

Typical performance measures in the runtime analysis of stochastic search algorithms are (among others):

- The probability $\mu_t = Pr\{x_t^{curr} \in S^*\}$ that the current solution in iteration t is optimal. He and Yu [57] (cf. also [111]) call $1 - \mu_t$ the *convergence rate*.
- The expected value or the distribution of the *first hitting time* (FHT) T_1, defined by $T_1 = \min\{t \geq 1 : x_t^{curr} \in S^*\}$.
- The expected value or the distribution of the time until a solution with a relative cost deviation from the optimum less than some ε has been found.

The measures above relate to a single given problem instance, say, a fixed distance matrix in the case of a TSP. In order to obtain more general information, one is usually rather interested in the behavior of A for a *class* of problem instances. In *complexity analysis*, all instances of a given problem of a certain *size* n are considered (say, all $[n \times n]$ distance matrices in the case of a TSP), where a suitable measure for instance size is applied. Then, the dependence of a fixed performance measure on n is studied. Since algorithm A has a different expected first hitting time for each instance of size n, some sort of aggregation is necessary. The two most important options for aggregation are to consider either the *worst case* performance

[3] To ask, say, for the expected time until an optimal solution is first hit without being sure that the optimum will be reached, is as meaningless as to ask: "How much training time would it take on average for a randomly selected person to win an olympic gold medal?" Also by being satisfied with an approximate solution of a certain minimum quality (call it the "silver medal") instead of the optimal solution, one does not escape this difficulty.

over all instances of size n, or the *average case* performance, given some probability distribution on the set of instances of size n.

16.4.1 Some Methods for Runtime Analysis

Each metaheuristic field has developed some specific techniques for analyzing computation times on selected optimization problems. However, a few general methods that turned out to be successful for more than one metaheuristic algorithm can be identified. Below, we shortly outline four of these methods. The reader is also referred to [5, 44, 81] for more details.

(1) Markov Chain Theory As noted in Sect. 16.2, the process (M_t) is a Markov process. In cases where the memory content M_t can only take finitely many values, the state space for this process is finite, i.e., (M_t) is a (homogeneous) *Markov chain*. An example are GAs, where M_t contains a population of solutions $x \in S$. In the probabilistic literature, much is known about Markov chains, and some results can be exploited for the analysis of the corresponding stochastic search algorithms. Following He and Yao [55], let us suppose, e.g., that by construction of A, states of M_t containing an optimal solution are never left again during the process (they are "absorbing states"), whereas other states have a probability larger than zero of being left in the next iteration (they are "transient states"). Let $\mathbf{A}$ and $\mathbf{T}$ denote the set of absorbing states and transient states, respectively, and let $j = |\mathbf{A}|$ and $k = |\mathbf{T}|$. Giving the j states in $\mathbf{A}$ the lowest and the k states in $\mathbf{T}$ the highest indices, the probability transition matrix P of the Markov chain (M_t) can be decomposed in the form

$$P = \begin{bmatrix} I_j & 0 \\ R & T \end{bmatrix},$$

where I_j is the $[j \times j]$ identity matrix, 0 is the $[j \times k]$ matrix with all elements equal to zero, and R and T are $[k \times j]$ and $[k \times k]$ matrices, respectively. He and Yao [55] show by direct application of a classical Markov chain result that the vector $\mathbf{m}$ whose ith component is the expected first hitting time m_i of the set of absorbing (i.e., optimal) states when starting from transient state i, is given by

$$\mathbf{m} = (I_k - T)^{-1}(1, \ldots, 1)^t,$$

where I_k is the $[k \times k]$ identity matrix. In principle, this would allow the computation of expected first hitting times, but the matrix $I_k - T$ is usually difficult to invert. Thus, the result can only be applied in cases where P has some special form (see, e.g., [81]). For other examples of the application of Markov chain theory, see, e.g., [22, 113]. In the last years, Markov chain approaches are often combined with drift analysis (see below).

(2) Level Sets This method evolved in papers on the analysis of evolutionary algorithms (EAs) such as [13] or [20]. It tries to circumvent the state-space explosion

for growing n, unavoidable in the direct application of the Markov chain approach, by grouping solutions into classes, where the fitness values are used as a natural criterion for defining the classes. Certain ranges of the fitness function ("levels") correspond to certain subsets ("level sets") of the search space S. The level sets have to be ordered in such a way that if $x \in A_j$ and $y \in A_k$ for two level sets A_j and A_k with $j < k$, it must always hold that $f(x) < f(y)$. In the easiest cases to analyze, the stochastic search algorithm A never returns to a level set corresponding to a lower fitness value after it has already visited a level set corresponding to a higher fitness value. For example, this monotonicity property is satisfied if A relies on the best-so-far solution x_t^{bsf} for the update from (M_t, L_t^+) to (M_{t+1}, L_{t+1}^+), since x^{bsf} can never decrease for increasing t. Now, if it is possible to determine a lower bound on the probability that the process jumps from some level j to a higher level $k > j$, an upper bound for the expected staying time in level j can be derived, and from those bounds, one can obtain an upper bound for the time until the highest (i.e., optimal) level is reached.

This idea has turned out to be fruitful for runtime analysis purposes not only in the field of EAs, but also in the ACO field (see, e.g., [44, 45, 49]). In the PSO field, the level-set method has been applied by Sudholt and Witt [98]. For an extension of the method using the concept of *potential functions*, see [108].

(3) Drift Analysis Drift analysis derives from martingale theory and has been applied for the analysis of SA (see [89]) and later for EAs (see, e.g., [54, 56]). Consider again the Markov process (M_t) and suppose that x_t^{curr} can be derived directly from M_t, i.e., $x_t^{curr} = x^{curr}(M_t)$. (If the information in L_t^+ is also required for getting x_t^{curr}, the process (M_t, L_t^+) must be considered instead of (M_t).) Based on x_t^{curr}, a *distance* $V(M)$ between state M and the set of states supporting optimal solutions may be defined. For example, one may set $V(M) = |f(x^{curr}(M)) - f^*|$, where f^* is the objective function value of the optimal solution. The one-step *mean drift* in state M is defined as the conditional expectation

$$E(V(M_t) - V(M_{t+1}) \,|\, M_t = M) = V(M) - \sum_{M'} P(M, M') V(M'),$$

where $P(M, M')$ is the transition probability from state M to state M'. If the mean drift is always zero, the process $V(M_t)$ is a martingale, which means that an optimal solution can only be found by chance. Hopefully, however, the drift generated by a stochastic search algorithm is positive, such that there is a tendency of the process to approach the set of optimal solutions.

He and Yao [54] show that from a lower bound on the mean drift, an upper bound on the expected first hitting time can be derived: If the mean drift in state M is larger than or equal to some constant $c_{low} > 0$ for any M with $V(M) > 0$, then the expected first hitting time after start in state M_1 satisfies $E(T_1 \,|\, M_1) \leq V(M_1)/c_{low}$. With the help of this and similar lemmas, the behavior of some EAs on simple test functions has been successfully analyzed. The generality of the formalism shows that drift analysis should be applicable in principle to every type of stochastic search algorithms.

In the last years, the application of drift analysis for obtaining runtime results has been considerably refined. For example, Oliveto and Witt [80] recently combined drift analysis, potential functions and the theory of submartingales in an investigation of the so-called Simple Genetic Algorithm, showing that already for the OneMax fitness function (discussed in Sect. 16.4.3 below), this algorithm requires exponential optimization time with overwhelming probability.

(4) Stochastic Approximation In some cases, where the process (M_t) itself appears too difficult for a mathematical analysis, one may try to obtain asymptotic approximations to this process for limiting cases concerning special parameter values. An example is given in [43, 45], where the behavior of the *Ant System* variant of ACO, developed by Dorigo et al. [18], is analyzed on simple test problems for a small learning rate ρ for the pheromone update (ρ is usually called "evaporation rate" in the ACO literature). In Ant System, the solution quality achieved in iteration t can also decrease compared to iteration $t-1$. Therefore, the level-set method is not applicable to this algorithm, contrary to some variants of MAX-MIN-Ant-System. However, letting ρ become small allows the application of the theory of slow learning that has been developed early in the learning literature (see [79]). As stated in Sect. 16.2, the memory M_t in ACO contains a vector of pheromone values. In the limiting case $\rho \to 0$, the dynamics of this vector becomes deterministic and can be described by a system of differential equations. Similar approaches have been pursued by Purkayastha and Baras [86] and by Paul and Mukhopdahyay [84].

Stochastic approximation techniques of this type may also be helpful for the analysis of other stochastic search algorithms where the memory content M_t lies in a continuous state space, e.g., EDAs or PSO. Indeed, in one of the first articles using an approach of this type, Gonzales et al. [37] refer to the analysis of PBIL, which is a special EDA.

16.4.2 Instance Difficulty and Phase Transitions

The methods presented in Sect. 16.4.1 analyze a stochastic metaheuristic A for a special problem instance (S, f). As noted at the beginning of Sect. 16.4, the topic of interest is typically not the behavior of A for a single instance, but for a class of instances, say the instances of size n of a given CO problem. The class may contain instances with completely different properties. Thus, the concepts of worst-case and of average-case analysis come into play.

In the case of some simple problems such as Generalized OneMax, which will be described in Sect. 16.4.3, the degree of difficulty is the same for all instances of size n. This is not true anymore for most CO problems found in applications. However, it seems that the degree of difficulty is usually not completely "scattered" among the instances, but often depends on some characteristic parameters of instances which are called *control parameters* or *order parameters*. The seminal paper by Cheesman et al. [15] has shown experimentally that for some fundamental

NP-hard combinatorial *decision* problems like k-SAT, Hamilton Circuits or Graph Coloring, different regions of the set of instances, such as "underconstrained" or "overconstrained" regions, must be distinguished; their boundary is defined by a critical value α_c of a control parameter, and the probability of the existence of a solution with the required properties changes abruptly from near zero to near one when crossing the boundary. The larger the instance size n, the sharper is the transition. By analogy to phenomena in physics like the melting of ice, this behavior is called *phase transition*. The computation time required for solving the problem is typically high near the phase transition and low for control parameter values far from α_c ("easy–hard–easy pattern"); sometimes, also "easy–hard" or "hard–easy" patterns are found.

Similar phase transition phenomena have also been observed in NP-hard combinatorial *optimization* problems, e.g., number partitioning [35], resource-constrained project scheduling problems [58], TSPs [112], independent-set problems [6], Max k-SAT [1], vertex-cover problems [53], and others—problems that form the natural range of application for stochastic search algorithms.

Whereas Cheesman et al. [15] use computational experiments for investigating the phase transition phenomenon in CO, essential progress in the theoretical understanding of this phenomenon has been achieved during the last decade in the physics literature, especially by applying concepts from *statistical mechanics* to CO. Martin et al. [71] and Monasson [75] give an introduction to this field. Statistical-mechanics investigations of CO problems usually start with the *Boltzmann distribution*[4] on the search space S, which is given by $p(x) = (1/Z_T)\exp(-f(x)/T)$, where $x \in S$ is a solution, $p(x)$ is the probability of x, f is the cost function (called *energy* in the physics literature), the parameter $T \geq 0$ is called *temperature*, and the normalization factor $Z_T = \sum_{y \in S} \exp(-f(y)/T)$ is called the *partition function*. The two boundary cases $T = \infty$ and $T = 0$ produce a uniform distribution on S, resp. a distribution that is concentrated on the set of global optima, the so-called *ground states*. The crucial idea is that by letting T tend toward zero and by investigating the partition function Z_T at this asymptotic limit, information on global optima is obtained, in particular information on the optimal cost function value f^* ("ground-state energy") or on the number of global optima.

In order to get from single problem instances to instance classes, the quantities derived from the partition function are averaged over the distribution of instances within the class of interest. As an example, consider the Number Partitioning Problem (NPP) with n items, the weights of which are represented by b-bit integers. A solution x consists in a partition of the set of given items into two subsets, and the cost function is the absolute difference between the total weights of the two subsets. Here, the ratio b/n turns out as the relevant control parameter. The statistical-mechanics approach predicts a phase transition around $b/n \sim 1$, with an exponentially growing search cost for b large compared to n, and polynomially growing cost for b small compared to n (see, e.g., [73]). This is in good agreement with exper-

[4] The relevance of this distribution in the field of stochastic search is also underlined by the fact that one of the oldest general-purpose stochastic search techniques, namely SA, approximates at each fixed temperature level T the corresponding Boltzmann distribution.

imental results. Recent work has focused on obtaining empirical insights into the changes in some properties such as number of local optima, plateaus or basin sizes when going through a phase transition. For the above-mentioned example of the NPP, e.g., Alyahya and Rowe [4] have presented such results.

From a practical point of view, the results on phase transitions indicate that for testing or tuning a metaheuristic algorithm, a suitable choice of the instance distribution is very important. In particular, it does not make too much sense to mix instances from the "easy" and "hard" regions, since the last will dominate the average behavior, which may mask the information that can be obtained for the easier instances.

16.4.3 Some Notes on Special Runtime Results

For reasons that will be discussed in Sect. 16.6, it is rather unlikely that for a stochastic search algorithm, universal positive runtime results (i.e., results valid for all CO problems predicting computation times of practical interest) can be obtained. Therefore, the promising way is to investigate different problems separately from each other, starting with very simple ones in order to develop useful analytical techniques, and successively progressing toward the hard CO problems found in applications.

Due to the necessity of studying runtime issues separately for single problems, the literature on analytical runtime results for stochastic search heuristics is rather dispersed. An overview would be beyond the scope of this chapter. Therefore, we focus on a few key issues. Classical results are recalled in the survey [81], which addresses the evolutionary algorithms field excluding the swarm-intelligence metaheuristics ACO and PSO, and the survey concerning ACO provided in [44]; some more recent results can be found in [5].

Typical simple problems investigated in the literature consist of artificially constructed test functions, usually pseudo-Boolean functions, i.e., functions mapping the set $S = \{0,1\}^n$ of binary strings $x = (x_1, \ldots, x_n)$ of length n into the reals. Examples are the OneMax fitness function $f(x) = \sum_{i=1}^{n} x_i$, the LeadingOnes fitness function $f(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j$, or the Needle-in-a-Haystack (NIAH) fitness function $f(x) = \prod_{j=1}^{n} x_j$. These three functions (instances) can be generalized to problems (classes of instances). For example, the Generalized OneMax problem contains the fitness functions $n - d_H(x,x^*)$, with d_H denoting the Hamming distance and $x^* \in S$ being an arbitrary fixed solution. (The OneMax function is the special case where $x^* = (1, \ldots, 1)$.) Also more general classes have been successfully analyzed in the literature. Droste et al. [20] have shown that for all *linear* pseudo-boolean functions, the expected first hitting time $E(T_1)$ of the simple $(1+1)$ EA grows as $\theta(n \log n)$ in the instance size n. In [104], some results on *quadratic* pseudo-boolean functions have been derived; this class already contains NP-hard optimization problems. Also for more complex EAs and for other stochastic metaheuristics, results concerning pseudo-boolean functions have been proved in the meantime (see the cited surveys).

Part of the literature analyzes the behavior of stochastic search algorithms on practically relevant problems from the complexity class P, i.e., problems for which polynomial-time solution algorithms exist. Such problems are good benchmarks for testing a metaheuristic algorithm which should be able to solve them by requiring only a low computational overhead compared to a problem-specific algorithm. In particular, sorting problems (e.g., [90]), maximum matching problems (e.g., [36]), and minimum spanning tree problems (e.g., [76, 78]) have been analyzed in an EA or ACO context. As expected, the investigated metaheuristics perform worse than the respective "tailored" algorithms, but they usually remain efficient in the sense that only polynomially growing runtime is required.

Very few works exist that analyze stochastic metaheuristics on NP-hard problems. Witt [107] investigates the behavior of the $(1+1)$ EA on a variant of the NPP (see Sect. 16.4.2) for which a fully polynomial approximation scheme exists. Within $O(n^2)$ steps, the $(1+1)$ EA finds a solution that is at least $(4/3)$-approximate. For the maximum clique problem on random planar graphs, Storch [94] proves that SA with constant temperature finds an optimal solution in linear time with overwhelming probability, while the $(1+1)$ EA needs $\theta(n^6)$ iterations. Also Wei and Dinneen [105] investigate the clique problem, comparing two fitness function choices. For the vertex cover problem, Friedrich et al. [29] show that the $(1+1)$ EA can produce arbitrarily poor solutions, whereas the evolutionary multi-objective optimizer SEMO performs sufficiently well. The poor performance of the $(1+1)$ EA can also be remedied by applying multistarts, as Oliveto et al. [82] demonstrate.

The issue of the possible advantage of random multistart also raises another interesting question. Whether or not random multistart is beneficial depends on the *distribution* of the first hitting time T_1. Therefore, results that do not only determine the expected value $E(T_1)$, but the entire distribution of the random variable T_1, would be very useful. Only a few results of this type seem to exist. We give two examples. Garnier et al. [32] show that for the first hitting time $T_1(n)$ of $(1+1)$ EA on a OneMax instance of size n, the re-normalized value $(T_1(n) - en\log n)/n$ converges in distribution to $-e\log Z + C$ as $n \to \infty$, where Z is exponentially distributed with parameter $\lambda = 1$, and C is a constant. Ladret [69] proves that for the first hitting time $T_1(n)$ of the $(1+1)$ EA on a LeadingOnes instance, the re-normalized value $(T_1(n) - mn^2)/n^{3/2}$ with $m = (e-1)/2$ converges in distribution to a normal distribution with mean 0 and variance $3(e^2-1)/8$.

Runtime complexity results are not an end in itself; ideally, they should guide the development of new and more efficient algorithms. In [17], Doerr et al. use runtime analysis to design a new crossover-based genetic algorithm that is asymptotically faster on OneMax than all previously known EAs.

16.5 Parameter Choice

One of the most important questions for the application of a metaheuristic algorithm is how its parameters should be chosen in order to obtain a good algorithmic performance for the application case at hand. Considering the functions g and h of the

generic algorithm of Sect. 16.2, we may distinguish between *sampling parameters* contained in g (they govern the distribution of the sample points in L_t), and *learning parameters* contained in h (they determine the type and amount of influence of the fitness values observed in the sampled trial points on the new memory content M_{t+1}). Examples of sampling parameters are mutation rate and crossover rate in GAs. Examples of learning parameters are temperature in SA or the learning rates used in ACO and in some EDAs, respectively.

A first question in this context is whether it is better to keep parameters constant during the optimization run or whether they should be changed dynamically. There are good empirical and theoretical arguments for the second alternative. Convergence results for more than one stochastic search algorithm are based on dynamic parameter schemes. For example, the classical convergence results [51] for SA require that the temperature parameter T is gradually decreased. Similarly, in [40], one of two indicated options for obtaining convergence of an ACO algorithm consists in gradually reducing the learning rate ρ. It seems that such a dynamic management of a central parameter of a stochastic search algorithm is a key instrument for achieving an exploration-exploitation balance.

Despite this intuitive consideration, it is surprisingly hard to verify the benefits of a dynamic parameter scheme through a rigorous demonstration that performance measures, such as the expected first hitting time, can be improved if the parameter values are *not* kept constant. For example, in the SA literature there has been a long discussion about the question "to cool or not to cool?": Is it really advantageous to decrease T during the optimization process, as theoretical convergence results suggest, or can the same performance be achieved by applying the so-called Metropolis algorithm which preserves a fixed, constant temperature T? At least for some practically occurring optimization problems, the former seems to be the case: Wegener [103] showed that for the minimum spanning tree problem, SA outperforms Metropolis.

In cases where there is no reason suggesting that a gradual reduction of the basic parameter of a stochastic search algorithm may be beneficial, we may still be interested in knowing whether it is better to keep the parameter at a fixed value, or to let it oscillate in some way in order to give the process a higher degree of variability. Jansen and Wegener [65] investigate this question analytically for the $(1+1)$ EA, applied to simple test functions such as OneMax and LeadingOnes (see Sect. 16.4.3). It turns out that the static variant where the parameter under consideration, the mutation probability p of the $(1+1)$ EA, is fixed to a constant, is better for some test functions than the dynamic variant where p is cyclically changed, and worse for some other test functions. The choice between the static and the dynamic scheme for a specific test function can make the difference between polynomial and exponential runtime.

Doerr and Doerr [16] analyze a certain dynamic rule for step size adaptation of a specific GA and show its superiority over any *static* version of this GA as well as its asymptotic optimality among adaptive parameter choices.

Apart from the question of whether or not parameters should be changed dynamically *during* the process, implementers of metaheuristics are always confronted with

the question of how the parameter values should be adapted to properties of specific problem instances, in particular the instance size n.

Let us give an example from the ACO domain showing how analytical results can help to get insight into this issue (for details, cf. [44]). The first investigations about the runtime of certain ACO variants [45, 77] seemed to indicate that for the Generalized OneMax problem, one has to apply relatively high values of the learning rate ρ to obtain the favorable expected first hitting time of order $\theta(n \log n)$ which is already known for the $(1+1)$ EA. In [49], it is demonstrated that a natural ACO algorithm of the MAX-MIN-Ant-System type can solve Generalized OneMax within expected time of order $\theta(n \log n)$ also for a *small* value of ρ independent of the problem size n. Similar results are obtained for the LeadingOnes problem. More than that: As soon as one goes from a fitness function giving "guidance" to the search process, as it is provided by OneMax or LeadingOnes, to a fitness function where parts of the optimal solution have to be identified by trial-and-error rather than by the guidance provided by the neighborhood structure, it becomes *essential* for the efficiency of the search process to choose ρ small enough. Consider, e.g., a combination of the functions OneMax and NIAH presented in Sect. 16.4.3 and defined by $f(x) = \left(\prod_{i=1}^{k} x_i\right) \cdot \left(\sum_{i=k+1}^{n} x_i + 1\right)$. For the maximization of this function, the correct bits on the first part of length k of the string must be found by trial-and-error (this is the NIAH part), and the remaining $n-k$ bits are optimized as for a OneMax problem. It is shown in [49] that this problem can only be solved efficiently if the learning rate ρ is decreased with increasing problem size n, but not too fast: For $k = \log_2 n$, a scheme of order $\theta(n^{-3})$ is suitable to obtain polynomial expected first hitting time. On the other hand, both $(1+1)$ EA and ACO with constant ρ require exponential expected time.

Another example is the following. In [98], Sudholt and Witt show that for the Binary PSO algorithm, keeping the (usually applied) bound v_{max} on the velocity of the particles fixed when increasing the instance size n leads to an extreme decline in performance. Scaling v_{max} to n by the function $v_{max} = \ln(n-1)$ considerably improves the runtime behavior on the considered test functions.

16.6 No-Free-Lunch Theorems

Looking at the co-existence of a considerable number of stochastic metaheuristics, a natural question would be to ask which is the "best" of them. In more specific terms: Can we derive a *universal* result stating that for all CO problems, some stochastic search algorithm A_1 always performs better than some other stochastic search algorithm A_2? Results of this type would be extremely valuable by simplifying the complex landscape of metaheuristics, but this hope broke down when Wolpert ad Macready [109] published their famous *No-Free-Lunch* (NFL) theorems for optimization. Basically, they state that when averaging over all possible fitness functions, no black-box search algorithm (may it be deterministic or stochastic) can be better than straightforward random search.

Before discussing this surprising result in more detail, we have to formulate the setting for which it holds in precise terms. It is not an essential restriction to assume that in our formulation (16.1) of a CO problem, the range of the function f is some *finite* subset Y of the set of reals: we may simply restrict the range to the image of the finite set S under f. Clearly, for fixed S and Y, there exist $|Y|^{|X|}$ different mappings (fitness functions) $f : S \to Y$. Assume that each of them has the same probability. Furthermore, let us restrict ourselves to search algorithms A (they can be deterministic or stochastic, i.e., the functions g and h introduced in Sect. 16.2 can depend on the random influences z and z' or not) with the property that the list L_t always contains only sample points $x \in S$ that have *not yet been visited* in previous iterations—in other words, with the property that the sets L_t are disjoint for $t = 1, 2, \ldots, t_{max}$, where t_{max} is the iteration in which algorithm A terminates. (Of course, this is a strong assumption, since it assumes that A stores information on already visited sample points in the memory M_t; its consequences, especially for *stochastic* search, where memory is saved to some extent by re-sampling, have not been fully investigated up to now.)

It is rather clear that under these circumstances, the fitness values of the sample points in S that have already been visited before some iteration t do not give any information on the fitness values of the sample points in S that have not yet been visited. Let us denote the set $\bigcup\{L_u | u < t\}$ of already visited points (solutions) by $S_v(t)$, so that $S \setminus S_v(t)$ is the set of yet unvisited points. As we assumed a uniform distribution on the set of all possible fitness functions f, the function values $f(x)$ for $x \in S \setminus S_v(t)$ are *independent* from the (observed) values for $x \in S_v(t)$. Therefore, no matter which rule algorithm A applies to determine L_t, information on the fitness values in $S_v(t)$ gathered in iterations $u = 1, \ldots, t-1$ does not provide any hint about the way to explore the yet completely unknown domain $S \setminus S_v(t)$. As a consequence, every rule is equally efficient on average; in particular, it is neither more efficient nor less efficient than random search. To each fitness function f for which A performs better than random search, there is another fitness function for which it performs worse.[5]

Of course, this result does not imply that on a special given *problem*, every stochastic search algorithm has the same performance. However, it seems that the NFL theorems force us to investigate search algorithms separately for each problem, because if A_1 dominates A_2 on some problem P_1, there must be another problem P_2 where A_2 dominates A_1. (For a recent discussion on the NFL theorems and their consequences for metaheuristics, see [61].)

[5] There seem to be close relations between NFL theorems and the well-known philosophical *induction problem* which also plays a role in AI approaches to inductive reasoning. Suppose that the evaluation of $f(x)$ for solution $x \in S$ is not done by an algorithmic computation, but rather by the observation of some real-world system (say, x is a control vector for a chemical plant, and $f(x)$ is the observed value of an outcome variable). Then the "NFL insight" that there is no logical argument for the observations $f(x_1), \ldots, f(x_{t-1})$ of some sample solutions $x_1, \ldots, x_{t-1}$ to provide any information on $f(x_t)$ for the sample solution $x_t \notin \{x_1, \ldots, x_{t-1}\}$, basically amounts to the intriguing claim by David Hume that we do not have any logical justification for the step called "inductive conclusion", although this step is essential in science as in everyday life.

Some researchers have drawn rather radical implications from the NFL theorems, questioning the field of metaheuristics as a whole. For example, Whitley and Watson [106] report that one extreme reaction is to conclude that there are no effective general-purpose search methods at all. Early on, there has already been a resistance against such over-interpretations, and authors have begun to investigate the limitations of the NFL theorems. Their arguments proceed mainly along two lines:

1. *Complexity Issues.* Whereas the NFL theorems hold for the set of *all possible* fitness functions, this set is usually not encountered in practice when solving optimization problems. Instead, the objective functions in classical CO problems have comparably low Kolmogorov complexity (KC). Droste et al. [19] show with an example that NFL theorems do not need to hold in classes of functions with restricted complexity, and that "intelligent" search algorithms are able in this context to outperform random search. English [23, 24] demonstrates that for search spaces S of medium to large size, almost all functions $f : S \to Y$ are "random" (in the sense of having a high KC), and he argues that random functions do not pose *practical* problems for heuristic optimization, because simple optimizers already discover good solutions quickly for them. On the other hand, "hard" problems are rare and therefore not represented adequately by the average-case consideration of the NFL theorems. Further results on the relation between KC and NFL theorems are presented, e.g., in [11].
2. *Influence of Fitness Landscape Properties.* Igel and Toussaint [62, 63] prove a sharpened NFL theorem giving a sufficient *and* necessary condition (closedness of the set of fitness functions under permutation) for the NFL theorem to hold. Under this condition, they show that if a non-trivial neighborhood structure on S has an influence on the fitness, the NFL theorem does *not* hold. In particular, a set of fitness functions satisfying certain steepness constraints entail that the assumptions of the NFL theorem are not satisfied. (Suitable steepness constraints may, e.g., exclude a case where by moving from a solution to an immediate neighbor solution, the cost function jumps from a global maximum to a global minimum.) Therefore, most practical applications do not fall under the NFL verdict. A similar conclusion is drawn in [68].

One may be relieved about the fact that NFL results do not have the consequence that developing efficient metaheuristics is a futile goal, but one should not miss the message: Since complexity properties are hard to deal with in applications (KC is not computable!), it seems that only the structure of the fitness landscape may "give us a free lunch" when applying general-purpose search algorithms. This leads us to the topic of fitness landscape analysis.

16.7 Fitness Landscape Analysis

A fitness landscape is formally defined as a triple (S, d, f), where S and f are the search space and fitness function, respectively, and d is a distance function on S assigning to each pair (x, x') of solutions $x, x' \in S$ a nonnegative integer distance

[74, 91]. (If a neighborhood structure on S—as used by ILS, SA or VNS—is given, d is derived in a natural way as the shortest distance in the neighborhood graph. Conversely, given d, solutions x, x' with $d(x,x') = 1$ are considered as neighbors.) In the literature, several quantities characterizing properties of the fitness landscape that are relevant for search algorithms have been defined (see, e.g., [74, 85, 87]). For the sake of shortness, let us restrict ourselves to two examples:

- The *fitness distance correlation* (FDC) is defined as

$$\rho(f, d_{opt}) = cov(f, d_{opt}) \,/\, [\sigma(f)\,\sigma(d_{opt})],$$

  where d_{opt} is the distance of a solution to the nearest optimal solution, while *cov* and σ denote covariance and standard deviation, respectively. For a maximization problem, a value of $\rho(f, d_{opt})$ near the minimum possible value of -1 indicates that the fitness is ideally correlated with the distance to the optimum solution; such landscapes are easy for stochastic local search algorithms or GAs. On the other hand, a value of $\rho(f, d_{opt})$ around 0 makes the problem harder, and a value near 1 indicates that the problem is "deceptive".
- The *random walk correlation function* is defined as the average of

$$r(s) = \frac{1}{\sigma^2(f)(m-s)} \sum_{t=1}^{m-s} (f(x_t) - \bar{f})\,(f(x_{t+s}) - \bar{f}),$$

  where $x_1, \ldots, x_m$ is a sampled random walk on the neighborhood graph of S, $\sigma^2(f)$ denotes the variance of the fitness, and $\bar{f}$ is the mean fitness.

Experimentally, a considerable effect of fitness landscape measures as those above on the efficiency of stochastic search algorithms has been observed. This effect has been exploited to improve algorithms, e.g., in [26, 59].

One of the most interesting parameters in fitness landscape analysis, and also a crucial parameter for the performance of stochastic local search algorithms, is the number N_{loc} of local optima. Fitness landscapes with a large number of local optima are "rugged" and hard for optimization. Reidys and Stadler [87] give a "correlation length conjecture" for the estimation of N_{loc} from the so-called correlation length $\ell = -[\ln(|r(1)|)]^{-1}$, where r is the random walk correlation function defined above. Empirical evidence supports this conjecture. Garnier and Kallel [31] describe a general technique for estimating N_{loc} by performing repeated local search with M random start solutions and by recording the number of times the found local optima are covered. Moreover, the methodology provides bounds on the search complexity for detecting all local optima. Eremeev and Reeves [25] are even able to determine confidence intervals for N_{loc}.

For some problems, the number of local optima can also be estimated analytically. For example, through the statistical-mechanics approach outlined in Sect. 16.4.2, Ferreira and Fontani [27] derive the expression $N_{loc} \sim 2.764 \cdot 2^n\, n^{-3/2}$ for the average number of local optima of the NPP problem (see Sect. 16.4.2) under

a uniform distribution model, which is in good agreement with simulation results. Considering that an NPP instance of size n has 2^n feasible solutions, we see that a simple ILS implementation will presumably not be very efficient in this case, compared to complete enumeration, except if one single local search run takes distinctly less than $O(n^{3/2})$ time.

16.8 Black-Box Optimization

The basic forms of most stochastic metaheuristics do not exploit any information about the specific problem instance (say, the distance matrix in a TSP), but only use information on the search space, including the neighborhood structure, as well as information on the specific problem type under consideration. In our generic framework, this scenario is defined by the condition that the functions g and h do not depend on the problem instance. Then, one may imagine that the algorithm A repeatedly calls a "black-box" procedure returning fitness values of given solutions x, but A does not "know" how these fitness values are determined. In this case, A is called a *black-box optimizer*.

From the viewpoint of a unified theory of stochastic search, it is interesting to investigate the potential of black-box optimizers independently from their specific algorithmic mechanisms. Recently, some articles have studied this issue. In [21], Droste et al. investigate upper and lower bounds for the expected first hitting times of stochastic black-box optimizers. The authors introduce a generic stochastic search algorithm "Black-Box Algorithm 1" which is essentially the generic algorithm of Sect. 16.2 with L_t restricted to a single element and M_t consisting of the entire search history, i.e., the sequence $(x_1, f(x_1), \ldots, x_{t-1}, f(x_{t-1}))$ of solutions visited before iteration t, together with their fitness values. In "Black-Box Algorithm 2", the size of the memory M_t is restricted by a bound $s(n)$ depending on the instance size n (which can be seen as a property of S; therefore, it does not violate the black-box restriction). The measure of performance is the expected optimization time in the worst case over all instances of size n.

For obtaining lower bounds on $E(T_1)$, the authors apply Yao's minimax principle [110], which can be stated as follows: The expected optimization time of a stochastic search algorithm A in the worst case over all instances is lower bounded by the expected value of the optimization time of an optimal deterministic search algorithm, where the expected value is taken with respect to an arbitrary instance distribution.

Droste et al. investigate sorting problems, for which they obtain linear lower bounds (and, depending on the fitness function used to evaluate the quality of a sort, linear or slightly super-linear upper bounds), on classes of simple functions such as the linear pseudo-boolean functions, and some more complex pseudo-boolean functions as well as special classes of unimodal functions. To give the flavor of the type of results, let us consider the Generalized OneMax example, for which

the lower bound $n/\log(2n+1)-1$ is derived in [21].[6] It is intuitively clear why we obtain a lower bound of order $O(n/\log n)$ here: Since there are $n+1$ different fitness function values, each call of the black-box procedure for determining the fitness of a solution x gives us $\log_2(n+1)$ bits of information. On the other hand, since there are 2^n alternatives for the optimal solution x^*, a total information of n bits is needed to identify x^*. Thus, an optimally designed search procedure will require about $n/\log_2(n)$ black-box calls.

Teytaud and Gelly [99] present lower bounds for black-box optimizers on problems with *continuous* search space and consider the scenario where only pairwise comparisons between fitness values are allowed to govern the search process instead of the overall information contained in the fitness value.

Another interesting perspective is presented by Borenstein and Poli [11, 12]; it relates NFL theorems, fitness landscape analysis and black-box optimization to each other. The authors argue that it is not sufficient to analyze the fitness landscape for itself; it is only by relating the latter to the operators used during the search that this information becomes relevant. A "proper" black-box optimizer should not have any a-priori preference for any regions of the search space, but rather select new sample points (in our notation: the elements of L_t) on the basis of their distance from already visited points. This requires that the applied search operators are in some sense consistent with the metric structure on S, which is described in [11] in algebraic terms.

Finally, let us mention that several practically applied variants of stochastic metaheuristics are not black-box optimizers, because they use information on the problem instance in addition to a black-box-type core mechanism. An example is the use of problem-specific heuristic values in ACO. We may call such algorithms *grey-box optimizers*, distinguishing them also from the "white-box" optimization techniques of mathematical programming (MP). Some metaheuristics, such as GRASP or Extreme Optimization, are inherently "grey-box". A special form of grey-box optimizers are hybrids between metaheuristics and MP approaches such as Local Branching [28], which have been termed *matheuristic* algorithms [14].

16.9 Stochastic Search Under Noise

In applications, it often happens that decisions are made under uncertainty, where certain parameters of an optimization problem are not deterministically known in advance. Often, it is possible to represent these parameters as random variables, by using an appropriate stochastic model. This leads to *stochastic optimization problems* where the objective function and sometimes also the constraints are disturbed

[6] Note that both EAs and ACO algorithms typically solve this problem in $O(n\log n)$ time [20, 49], which differs from the lower bound by a factor of order $O((\log n)^2)$. This overhead may partly be explained by the effort for re-sampling already visited solutions.

by "noise". In this section, we restrict ourselves to stochastic combinatorial optimization (SCO) problems of the following frequently occurring form, which is a natural extension of the deterministic CO problem (16.1):

$$\min E(f(x,\omega)) \text{ such that } x \in S. \tag{16.2}$$

Therein, E denotes the expectation operator, and ω is a random influence with a distribution given by the stochastic model of the problem (ω is not to be confounded with the random variables z and z' used by the stochastic search algorithm, see Sect. 16.2). As in the deterministic case, "min" can be replaced by "max". For example, consider the *stochastic total tardiness problem*, where a set of jobs $1,\ldots,n$ together with their due dates $d_1,\ldots,d_n$ are given. Each job has a processing time Y_i which is a random variable with known distribution. The objective is to find a sequential arrangement x of the n jobs such that the expected value of the sum of their tardiness values is minimized, where the tardiness of job i is $(C_i - d_i)^+$, with C_i denoting the completion time of job i. Note that $C_i = C_i(x,Y)$ depends both on the solution x and on the vector Y of random processing times. Here, ω can be considered as identical to Y, and $f(x,\omega) = \sum_{i=1}^{n}(C_i(x,\omega) - d_i)^+$.

In the literature on stochastic optimization, several methods have been developed to solve problems of the form (16.2). In particular, *metaheuristic* algorithms have also been applied in this field; surveys are given in [9, 46, 60]. A survey focusing on EAs can also be found in [66].

Basically, three different approaches followed by metaheuristic search algorithms under the black-box optimization paradigm[7] can be distinguished: (1) If possible, a procedure for the numerical computation of the expectation in (16.2) is implemented, and the problem is solved in the same manner as a deterministic CO problem, performing black-box calls of the numerical procedure to obtain fitness evaluations. Often, this requires a large amount of computation time or is even infeasible. (2) A sample of random instances for the uncertain parameters, distributed according to the given stochastic model, is generated as an approximation of the exact distribution; after that, large-scale optimization averaging over this sample is done. This is called a *fixed-sample* approach. (3) In a *variable-sample* approach, sampling and optimization are not two successive phases, but rather alternate over the iterations of the search algorithm. This allows the use of smaller sample sizes.

Note that a combination of these different approaches can be convenient for particular problems where the (small) error intrinsically affecting sampling methods is amplified by the characteristics of the problem: Consider a stochastic scheduling problem with the same settings than the total tardiness problem mentioned earlier, but with an objective function given by $f(x,\omega) = [Pr(C_i(x,\omega) \leq d_i)q_i - (1 - Pr(C_i(x,\omega) \leq d_i))e_i]$, where $Pr(\gamma)$ is the probability for event γ to happen, according to the given distribution, q_i is a reward gained if job i is expected to be completed before the deadline d_i, and e_i is a penalty paid in case the deadline d_i is not respected for job i. For such a problem,

[7] For approaches using "white-box" mathematical programming techniques such as the Integer L-Shaped Method, see, e.g., [34].

even small errors caused by sampling approaches may produce grossly distorted estimates of the objective function, due to jobs for which $|C_i - d_i|$ is close to 0: rewards or penalties can be erroneously attributed in such circumstances. A natural solution is to make a trade off between precision and computational speed by using sampling approaches when $|C_i - d_i|$ is greater than a safety threshold, and numerical approximation otherwise, for critical jobs. Notice that more complex strategies can also be devised [83].

General-purpose variable-sample SCO algorithms have been derived from certain metaheuristics such as SA [3, 33, 48], ACO [10, 41, 42] or VNS [50]. The structure of these algorithms is an extension of our generic scheme of Sect. 16.2. We only have to replace in step 2b of the generic algorithm the evaluation of the objective function values $f(x_i)$ by *sample average estimates* $\tilde{F}(x_i) = \sum_{v=1}^{N} f(x_i, \omega_v)$ approximating $F(x_i) = E(f(x_i, \omega))$, where the ω_v form a random sample for the uncertain parameter ω according to the given distribution. The sample size N does not need to be fixed over the iterations, but can be chosen as a function of M_t. Typically, N is gradually increased to improve the accuracy of the estimate.

Using this approach and using a suitable scheme for $N = N(M_t)$, convergence results for the previously mentioned modifications of SA, ACO or VNS are reported in [41, 48, 50] by generalizing known convergence results for the corresponding basic metaheuristics. Furthermore, work on runtime analysis of such algorithms has recently started [2, 30, 47, 97].

16.10 Stochastic Search and Robustness

Sometimes it is not possible to come out with a probability distribution for the parameters of a problem under uncertainty, as previously assumed in Sect. 16.9. This can happen either because the uncertain phenomena cannot be captured by a mathematical distribution, or because there is not enough data to identify a distribution. In such cases, one may rely on robust optimization (RO) [93], using, e.g., the so called *interval data* model [8]. An input information for a problem parameter affected by uncertainty corresponds to an interval defined by a lower and an upper bound. All values within such an interval are possible, but the underlying distribution of the values is considered unknown. Such a model is less precise than those based on stochastic information, but is much easier to handle from a computational viewpoint, and has proven to provide results of great interest for practitioners [7].

The first robust optimization approaches [93] were protecting the decision maker against the worst possible scenarios by taking the worst possible values for all uncertain parameters, from the decision maker point of view. Later, compromise solutions, which are less conservative and typically more practical, were considered. One of the most prominent approaches of this kind was proposed in Bertsimas and Sim [8] who presented robust optimization models where the decision maker can configure the degree of conservatism according to her/his needs. The corresponding robust optimization techniques, based on mathematical programming, provide op-

timal solutions to the model, but are often not suitable for real-life problems, due to their long running time on medium/large instances. Therefore, general-purpose robust heuristic algorithms have been derived as an extension of our generic scheme described in Sect. 16.2. The main idea is to plug a new objective function evaluator inside an algorithm that, given a solution, returns its robustness cost, taking into account both uncertainty and the degree of conservatism chosen by the decision maker. Such an evaluation is typically obtained by solving a small linear programming model, as originally described in [8]. It has been observed that the robust version of a metaheuristic algorithm typically takes about twice the time of a deterministic version to carry out the same number of iterations [100]. Although the SA [7] and ACO [100] metaheuristics have been presented in the OR domain, to the best of our knowledge, no formal convergence results are available at present for these algorithms.

16.11 Conclusions

In the past, metaheuristics have evolved in different scientific sub-communities, separated from each other to a certain extent. Although a strong tendency towards cross-linking can be observed between these sub-communities (see [72]), already resulting in considerable synergy effects as well as in the establishment of a joint experimental methodology, a common theoretical framework enabling an immediate exploitation of progress in one of the metaheuristic subfields by other subfields seems still to be lacking. Much work still has to be done for achieving a unified understanding of metaheuristic algorithms.

One of the key elements around which a holistic view of the different metaheuristic techniques may be organized is the role of *stochastic search* in most of them. The results cited in this chapter may indicate possible starting points for a process leading to a general theory of stochastic search, but this process still has to take place. Anyway, the many successes of particular metaheuristics in solving real-world problems should not lead the community astray from attempting to consider the "big picture" by looking at what single metaheuristic paradigms have in common.

As discussed in Sect. 16.6, it is not likely that one single metaheuristic will turn out as "superior" to the others and throw them out from the application fields. Rather, it may be anticipated that the current co-existence of different metaheuristics will prevail. This is desirable if we want to increase our understanding of the benefits of each metaheuristic through a common framework that will allow them to be compared.

Many important topics have been excluded in this chapter, such as stochastic search in (non-linear) continuous, multi-objective, or dynamic optimization. Other issues, like runtime analysis, are even less developed in these areas than in single-objective static CO, and many related open problems represent a challenge for future research.

References

1. D. Achlioptas, A. Naor, Y. Peres, Rigorous location of phase transitions in hard optimization problems. Nature **435**, 759–764 (2005)
2. Y. Akimoto, S. Astete-Morales, O. Teytaud, Analysis of runtime of optimization algorithms for noisy functions over discrete codomains. Theor. Comput. Sci. **605**, 42–50 (2015)
3. M.H. Alrefaei, S. Andradottir, A simulated annealing algorithm with constant temperature for discrete stochastic optimization. Manag. Sci. **45**, 748–764 (1999)
4. K. Alyahya, J.E. Rowe, Phase transition and landscape properties of the number partitioning problem, in *European Conference on Evolutionary Computation in Combinatorial Optimization* (Springer, Berlin, 2014), pp. 206–217
5. A. Auger, B. Doerr (eds.), *Theory of Randomized Search Heuristics: Foundations and Recent Developments*, vol. 1 (World Scientific, Singapore, 2011)
6. V.C. Barbosa, R.G. Ferreira, On the phase transitions of graph coloring and independent sets. Phys. A **343**, 401–423 (2004)
7. D. Bertsimas, O. Nohadani, Robust optimization with simulated annealing. J. Glob. Optim. **48**, 323–334 (2010)
8. D. Bertsimas, M. Sim, The price of robustness. Oper. Res. **52**, 35–53 (2004)
9. L. Bianchi, M. Dorigo, L.M. Gambardella, W.J. Gutjahr, A survey on metaheuristics for stochastic combinatorial optimization. Nat. Comput. **8**, 239–287 (2009)
10. M. Birattari, P. Balaprakash, M. Dorigo, The ACO/FRACE algorithm for combinatorial optimization under uncertainty, in *Metaheuristics – Progress in Complex Systems Optimization*, ed. by K. Doerner et al. (Springer, Berlin, 2006)
11. Y. Borenstein, R. Poli, Information perspective of optimization, in *Proceedings of the 9th Conference on Parallel Problem Solving from Nature*. Springer LNCS, vol. 4193 (2006), pp. 102–111
12. Y. Borenstein, R. Poli, Structure and metaheuristics, in *Proceedings of the Genetic and Evolutionary Computation Conference '06* (2006), pp. 1087–1093
13. P.A. Borisovsky, A.V. Eremeev, A study on performance of the $(1+1)$-evolutionary algorithm, in *Proceedings of the Foundations of Genetic Algorithms*, vol. 7 (Morgan Kaufmann, San Francisco, 2003), pp. 271–287
14. M.A. Boschetti, V. Maniezzo, M. Roffilli, A.B. Röhler, Matheuristics: optimization, simulation and control, in *International Workshop on Hybrid Metaheuristics* (Springer, Berlin, 2009), pp. 171–177
15. P. Cheesman, B. Kenafsky, W.M. Taylor, Where the really hard problems are, in *Proceedings of the IJCAI '91* (Morgan Kaufmann, San Francisco, 1991), pp. 331–337
16. B. Doerr, C. Doerr, Optimal parameter choices through self-adjustment: applying the 1/5-th rule in discrete settings, in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation* (ACM, New York, 2015), pp. 1335–1342
17. B. Doerr, C. Doerr, F. Ebel, From black-box complexity to designing new genetic algorithms. Theor. Comput. Sci. **567**, 87–104 (2015)
18. M. Dorigo, V. Maniezzo, A. Colorni, Ant System: optimization by a colony of cooperating agents. IEEE Trans. Syst. Man, Cybern. **26**, 1–13 (1996)
19. S. Droste, T. Jansen, I. Wegener, Perhaps not a free lunch but at least a free appetizer, in *Proceedings of the Genetic and Evolutionary Computation Conference '99* (1999), pp. 833–839
20. S. Droste, T. Jansen, I. Wegener, On the analysis of the $(1+1)$ evolutionary algorithm. Theor. Comput. Sci. **276**, 51–81 (2002)
21. S. Droste, T. Jansen, I. Wegener, Upper and lower bounds for randomized search heuristics in black-box optimization. Theory Comput. Syst. **39**, 525–544 (2006)
22. X. Du, L. Ding, About the convergence rates of a class of gene expression programming. Sci. China Inf. Sci. **53**, 715–728 (2010)
23. T. English, Optimization is easy and learning is hard in the typical function, in *Proceedings of the Congress in Evolutionary Computation '00* (2000), pp. 924–931

24. T. English, On the structure of sequential search: beyond "no free lunch", in *Proceedings of the EvoCOP '04*. Springer LNCS, vol. 3004 (2004), pp. 95–103
25. A.V. Eremeev, C.R. Reeves, On confidence intervals for the number of local optima, in *Applications of Evolutionary Computing*. Springer LNCS, vol. 2611 (2003), pp. 224–235
26. M. Eskandarpour, E. Nikbakhsh, S.H. Zegordi, Variable neighborhood search for the bi-objective post-sales network design problem: a fitness landscape analysis approach. Comput. Oper. Res. **52**, 300–314 (2014)
27. F.F. Ferreira, J.F. Fontanari, Probabilistic analysis of the number partitioning problem. J. Phys. A Math. Gen. **31**, 3417–3428 (1998)
28. M. Fischetti, A. Lodi, Local branching. Math. Program. Ser. B **98**, 23–47 (2003)
29. T. Friedrich, J. He, N. Hebbinghaus, F. Neumann, C. Witt, Approximating covering problems by randomized search heuristics using multi-objective models, in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation* (2007), pp. 797–804
30. T. Friedrich, T. Kötzing, M.S. Krejca, A.M. Sutton, The benefit of recombination in noisy evolutionary search, in *International Symposium on Algorithms and Computation* (Springer, Berlin, 2015), pp. 140–150
31. J. Garnier, L. Kallel, Efficiency of local search with multiple local optima. SIAM J. Discrete Math. **15**, 122–141 (2002)
32. J. Garnier, L. Kalel, M. Schoenauer, Rigorous hitting times for binary mutations. Evol. Comput. **7**, 45–68 (1999)
33. S.B. Gelfand, S.K. Mitter, Simulated annealing with noisy or imprecise measurements. J. Optim. Theory Appl. **69**, 49–62 (1989)
34. M. Gendreau, G. Laporte, R. Seguin, An exact algorithm for the vehicle routing problem with stochastic demands and customers. Transp. Sci. **29**, 143–155 (1995)
35. I.P. Gent, T. Walsh, Analysis of heuristics for number partitioning. Comput. Intell. **14**, 430–450 (1998)
36. O. Giel, I. Wegener, Evolutionary algorithms and the maximum matching problem, in *Proceedings of the 20th Annual Symposium on Theoretical Aspects of Computer Science* (2003), pp. 415–426
37. C. Gonzalez, J.A. Lozano, P. Larrañaga, Analyzing the PBIL algorithm by means of discrete dynamical systems. Complex Syst. **11**, 1–15 (1997)
38. C. Gonzalez, J.A. Lozano, P. Larrañaga, Mathematical modelling of discrete estimation of distribution algorithms, in *Estimation of Distribution Algorithms, A New Tool for Evolutionary Computation*, ed. by Larranaga et al. (Kluwer Academic Publishers, Boston, 2002), pp. 147–163
39. W.J. Gutjahr, A graph-based ant system and its convergence. Futur. Gener. Comput. Syst. **16**, 873–888 (2000)
40. W.J. Gutjahr, ACO algorithms with guaranteed convergence to the optimal solution. Inf. Process. Lett. **82**, 145–153 (2002)
41. W.J. Gutjahr, A converging ACO algorithm for stochastic combinatorial optimization, in *Proceedings of the 2nd Symposium on Stochastic Algorithms, Foundations and Applications*. Springer LNCS, vol. 2827 (2003), pp. 10–25
42. W.J. Gutjahr, S-ACO: an ant-based approach to combinatorial optimization under uncertainty, in *Proceedings of the 4nd International Workshop on Ant Colony Optimization and Swarm Intelligence*. Springer LNCS, vol. 3172 (2004), pp. 238–249
43. W.J. Gutjahr, On the finite-time dynamics of ant colony optimization. Methodol. Comput. Appl. Probab. **8**, 105–133 (2006)
44. W.J. Gutjahr, Mathematical runtime analysis of ACO algorithms: survey on an emerging issue. Swarm Intell. **1**, 59–79 (2007)
45. W.J. Gutjahr, First steps to the runtime complexity analysis of ant colony optimization. Comput. Oper. Res. **35**, 2711–2727 (2008)
46. W.J. Gutjahr, Recent trends in metaheuristics for stochastic combinatorial optimization. Cen. Eur. J. Comput. Sci. **1**, 58–66 (2011)
47. W.J. Gutjahr, Runtime analysis of an evolutionary algorithm for stochastic multi-objective combinatorial optimization. Evol. Comput. **20**, 395–421 (2012)

48. W.J. Gutjahr, G. Pflug, Simulated annealing for noisy cost functions. J. Glob. Optim. **8**, 1–13 (1996)
49. W.J. Gutjahr, G. Sebastiani, Runtime analysis of ant colony optimization with best-so-far reinforcement. Methodol. Comput. Appl. Probab. **10**, 409–433 (2008)
50. W.J. Gutjahr, S. Katzensteiner, P. Reiter, A VNS algorithm for noisy problems and its application to project portfolio analysis, in *Proceedings of the SAGA 2007 (Stochastic Algorithms: Foundations and Applications)*. Springer LNCS, vol. 4665 (2007), pp. 93–104
51. B. Hajek, Cooling schedules for optimal annealing. Math. Oper. Res. **13**, 311–329 (1988)
52. R.F. Hartl, A global convergence proof for a class of genetic algorithms. Technical Report, University of Vienna (1990)
53. A.K. Hartmann, W. Barthel, M. Weigt, Phase transition and finite-size scaling in the vertex-cover problem. Comput. Phys. Commun. **169**, 234–237 (2005)
54. J. He, X. Yao, Drift analysis and average time complexity of evolutionary algorithms. Artif. Intell. **127**, 57–85 (2003)
55. J. He, X. Yao, Towards an analytic framework for analysing the computation time of evolutionary algorithms. Artif. Intell. **145**, 59–97 (2003)
56. J. He, X. Yao, A study of drift analysis for estimating computation time of evolutionary algorithms. Nat. Comput. **3**, 21–35 (2004)
57. J. He, X. Yu, Conditions for the convergence of evolutionary algorithms. J. Syst. Archit. **47**, 601–612 (2001)
58. W. Herroelen, B. De Reyck, Phase transitions in project scheduling. J. Oper. Res. Soc. **50**, 148–156 (1999)
59. J. Humeau, A. Liefooghe, E.G. Talbi, S. Verel, ParadisEO-MO: from fitness landscape analysis to efficient local search algorithms. J. Heuristics **19**, 881–915 (2013)
60. L.M. Hvattum, E.F. Esbensen, Metaheuristics for stochastic problems, in *Wiley Encyclopedia of Operations Research and Management Science* (Wiley, Hoboken, 2011)
61. C. Igel, No free lunch theorems: limitations and perspectives of metaheuristics, in *Theory and Principled Methods for the Design of Metaheuristics* (Springer, Berlin, 2014), pp. 1–23
62. C. Igel, M. Toussaint, On classes of functions for which no free lunch results hold. Inf. Process. Lett. **86**, 317–321 (2003)
63. C. Igel, M. Toussaint, A no-free-lunch theorem for non-uniform distributions of target functions. J. Math. Model. Algorithms **3**, 313–322 (2004)
64. S.H. Jacobson, E. Yücesan, Analyzing the performance of generalized hill climbing algorithms. J. Heuristics **10**, 387–405 (2004)
65. T. Jansen, I. Wegener, On the analysis of a dynamic evolutionary algorithm. J. Discrete Algorithms **4**, 181–199 (2006)
66. Y. Jin, J. Branke, Evolutionary optimization in uncertain environments – a survey. IEEE Trans. Evol. Comput. **9**, 303–317 (2005)
67. J. Kennedy, R.C. Eberhart, A discrete binary version of the particle swarm algorithm, in *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics* (1997), pp. 4104–4109
68. G.J. Koehler, Conditions that obviate the no-free-lunch theorems for optimization. Inform. J. Comput. **19**, 273–279 (2007)
69. V. Ladret, Asymptotic hitting time for a simple evolutionary model of protein folding. J. Appl. Probab. **42**, 39–51 (2005)
70. L. Margolin, On the convergence of the cross-entropy method. Ann. Oper. Res. **134**, 201–214 (2005)
71. O.C. Martin, R. Monasson, R. Zecchina, Statistical mechanics methods and phase transitions in optimization problems. Theor. Comput. Sci. **265**, 3–67 (2001)
72. J.-J. Merelo, C. Cotta, Building bridges: the role of subfields in metaheuristics. SIGEVOlution **1**(4), 9–15 (2006)
73. S. Mertens, A physicist's approach to number partitioning. Theor. Comput. Sci. **265**, 79–108 (2001)
74. P. Merz, B. Freisleben, Fitness landscape analysis and memetic algorithms for the quadratic assignment problem. IEEE Trans. Evol. Comput. **4**, 337–352 (2000)

75. R. Monasson, Introduction to phase transitions in random optimization problems. Technical Report, Laboratoire de Physique Theorique de l'ENS, Paris (2007)
76. F. Neumann, I. Wegener, Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. Theor. Comput. Sci. **378**, 32–40 (2007)
77. F. Neumann, C. Witt, Runtime analysis of a simple ant colony optimization algorithm, in *Proceedings of the ISAAC '06*. Springer LNCS, vol. 4288 (2006), pp. 618–627
78. F. Neumann, C. Witt, Ant colony optimization and the minimum spanning tree problem. Theor. Comput. Sci. **411**, 2406–2413 (2010)
79. F. Norman, *Markov Processes and Learning Models* (Academic, New York, 1972)
80. P.S. Oliveto, C. Witt, Improved time complexity analysis of the simple genetic algorithm. Theor. Comput. Sci. **605**, 21–41 (2015)
81. P.S. Oliveto, J. He, X. Yao, Time complexity of evolutionary algorithms for combinatorial optimization: a decade of results. Int. J. Autom. Comput. **4**, 281–293 (2007)
82. P.S. Oliveto, J. He, X. Yao, Evolutionary algorithms and the vertex cover problem, in *Proceedings of the Congress on Evolutionary Computation CEC '07* (2007), pp. 1870–1877
83. V. Papapanagiotou, R. Montemanni, L.M. Gambardella, Sampling-based objective function evaluation techniques for the Orienteering Problem with Stochastic Travel and Service Times, in *Operations Research Proceedings 2014* (Springer, Cham, 2016), pp. 445–450
84. A. Paul, S. Mukhopadhyay, A frequency domain analysis on the deterministic modeling of the Ant System dynamics, in *Third International Conference on Computer, Communication, Control and Information Technology (C3IT)* (IEEE, Piscataway, 2015), pp. 1–6
85. E. Pitzer, M. Affenzeller, A comprehensive survey on fitness landscape analysis, in *Recent Advances in Intelligent Engineering Systems* (Springer, Berlin, 2012), pp. 161–191
86. P. Purkayastha, J.S. Baras, Convergence results for ant routing algorithms via stochastic approximation and optimization, in *Proceedings of the 46th IEEE Conference on Decision and Control* (2007), pp. 340–345
87. C.M. Reidys, P.F. Stadler, Combinatorial landscapes. SIAM Rev. **44**, 3–54 (2002)
88. G. Rudolph, Convergence Analysis of canonical genetic algorithms. IEEE Trans. Neural Netw. **5**, 96–101 (1994)
89. G.H. Sasaki, B. Hajek, The time complexity of maximum matching by simulated annealing. J. ACM **35**, 67–89 (1988)
90. J. Scharnow, K. Tinnefeld, I. Wegener, Fitness landscapes based on sorting and shortest path problems, in *Proceedings of the 7th Conference on Parallel Problem Solving from Nature* (2002), pp. 54–63
91. T. Schiavinotto, T. Stûtzle, A review of metrics on permutations for search landscape analysis. Comput. Oper. Res. **34**, 3143–3153 (2007)
92. G. Sebastiani, G.L. Torrisi, An extended ant colony algorithm and its convergence analysis. Methodol. Comput. Appl. Probab. **7**, 249–263 (2005)
93. A.L. Soyster, Convex programming with set-inclusive constraints and applications to inexact linear programming. Oper. Res. **21**, 1154–1157 (1973)
94. T. Storch, How randomized search heuristics find maximum cliques in planar graphs, in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation* (2006), pp. 567–574
95. T. Stützle, H.H. Hoos, MAX-MIN Ant System. Futur. Gener. Comput. Syst. **16**, 889–914 (2000)
96. T. Stützle, M. Dorigo, A short convergence proof for a class of ACO algorithms. IEEE Trans. Evol. Comput. **6**, 358–365 (2002)
97. D. Sudholt, C. Thyssen, Running time analysis of ant colony optimization for shortest path problems. J. Discrete Algorithms **10**, 165–180 (2012)
98. D. Sudholt, C. Witt, Runtime analysis of binary PSO, in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation* (2008), pp. 135–142
99. O.Teytaud, S. Gelly, General lower bounds for evolutionary algorithms, in *Proceedings of the 9th Conference on Parallel Problem Solving from Nature* (2006), pp. 21–31

100. N.E. Toklu, R. Montemanni, L.M. Gambardella, A robust multiple ant colony system for the capacitated vehicle routing problem, in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics* (2013), pp. 1871–1876
101. I.C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection. Inf. Process. Lett. **85**, 317–325 (2003)
102. D.E. Vaughan, S.H. Jacobson, H. Kaul, Analyzing the performance of simultaneous generalized hill climbing algorithms. Comput. Optim. Appl. **37**, 103–119 (2007)
103. I. Wegener, Simulated annealing beats metropolis in combinatorial optimization, in *Proceedings of the ICALP '05*. Springer LNCS, vol. 3580 (2005), pp. 589–601
104. I. Wegener, C. Witt, On the analysis of a simple evolutionary algorithm on quadratic pseudo-boolean functions. J. Discrete Algorithms **3**, 61–78 (2005)
105. K. Wei, M.J. Dinneen, Runtime analysis comparison of two fitness functions on a memetic algorithm for the clique problem, in *2014 IEEE Congress on Evolutionary Computation (CEC)* (2014), pp. 133–140
106. D. Whitley, J.P. Watson, Complexity theory and the no free lunch theorem. in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*, ed. by E.K. Burke, G. Kendall (Kluwer, Boston, 2005), pp. 317–399
107. C. Witt, Worst-case and average-case approximations by simple randomized search heuristics, in *Proceedings of the 22nd Annual Symposium on Theoretical Aspects of Computer Science*. Springer LNCS, vol. 3404 (2005), pp. 44–56
108. C. Witt, Runtime analysis of the $(\mu + 1)$ EA on simple pseudo-boolean functions. in *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, Seattle, Washington (2006), pp. 651–658
109. D.H. Wolpert, W.G. Macready, No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**, 67–82 (1997)
110. A.C. Yao, Probabilistic computations: towards a unified measure of complexity, in *Proceedings of the 17th IEEE Symposium on the Foundations of Computer Science* (1977), pp. 222–227
111. Y. Yu, Z.-H. Zhou, A new approach to estimating the expected first hitting time of evolutionary algorithms, in *Proceedings of the 21th National Conference on Artificial Intelligence*, Boston (2006), pp. 555–560
112. W. Zhang, Phase transitions and backbones of the asymmetric travelling salesman problem. J. Artif. Intell. Res. **21**, 471–497 (2004)
113. Y. Zhou, J. He, Q. Nie, A comparative runtime analysis of heuristic algorithms for satisfiability problems. Artif. Intell. **173**, 240–257 (2009)
114. M. Zlochin, M. Birattari, N. Meuleau, M. Dorigo, Model-based search for combinatorial optimization: a critical survey. Ann. Oper. Res. **131**, 373–379 (2004)