Chapter 6
Convergence Analysis of Metaheuristics

Walter J. Gutjahr

Abstract In this tutorial, an overview on the basic techniques for proving convergence of metaheuristics to optimal (or sufficiently good) solutions is given. The presentation is kept as independent of special metaheuristic fields as possible by the introduction of a generic metaheuristic algorithm. Different types of convergence of random variables are discussed, and two specific features of the search process to which the notion "convergence" may refer, the "best-so-far solution" and the "model", are distinguished. Some core proof ideas as applied in the literature are outlined. We also deal with extensions of metaheuristic algorithms to stochastic combinatorial optimization, where convergence is an especially relevant issue. Finally, the important aspect of convergence speed is addressed by a recapitulation of some methods for analytically estimating the expected runtime until solutions of sufficient quality are detected.

6.1 Introduction

Metaheuristic algorithms (see, e.g., the standard textbook [19]) are general-purpose solvers for optimization problems. Their use is prevalent in modern applications of computational intelligence because of their broad flexibility, their intuitive structures and their applicability even in the case of problem instances for which classical optimization techniques fail. A huge amount of knowledge about the *empirical* performance of diverse metaheuristics on numerous classes of optimization problems has been accumulated within the last two decades. Theoretical analysis lags behind this development. While some aspects of the performance of metaheuristics are already well-understood,

Department of Statistics and Decision Support Systems, University of Vienna, Vienna, Austria
e-mail: walter.gutjahr@univie.ac.at

V. Maniezzo et al., (eds.), *Matheuristics*, Annals of Information Systems 10,
DOI 10.1007/978-1-4419-1306-7_6,

some other issues still wait for an analytical treatment. One of the most basic issues is the question whether or not the current solutions proposed by a metaheuristic converge to an optimal or at least to a "sufficiently good" solution—and if yes, how fast this happens.

Especially the recent attempts of combining metaheuristic techniques with the well-established methods from mathematical programming (MP) to powerful hybrids give the convergence issue a new relevance. Researchers from the MP field are used to be sure that their algorithms reach an optimal solution within finite time. More than that, even a proof of optimality is delivered by an MP method, typically within a computation time that can be bounded from above (although the bound may be very large). In contrast to that, metaheuristics are not only unable to produce proven optimality; some often applied variants cannot even guarantee that an optimal solution (or a solution of a certain minimum quality) will be eventually found if an unlimited amount of computation time is invested. An algorithm embedding an MP component within a metaheuristic inherits this weakness.

Fortunately, however, convergence to the optimum *can* be shown for several variants (or parameterizations) of metaheuristic algorithms, and usually, convergence of an algorithm can even be ensured simultaneously for a very broad range of problems, e.g., for all combinatorial optimization (CO) problems. Identifying conditions that entail convergence provides a starting point also for the practically very relevant question of convergence speed.

In this tutorial, we give an introduction to the basic ideas of convergence proofs for metaheuristics. The scope will be restricted by two limitations: First, we shall focus on *global* convergence instead of converge to local optima. In particular, dynamical-systems approaches to the analysis of metaheuristics (cf. [20] or [60, 9]) are not addressed here because of their orientation towards local convergence. Secondly, we concentrate on discrete finite search spaces, i.e., on the CO case. There is a large amount of literature on convergence of search algorithms on continuous search spaces, let us only refer to [56, 59, 45] for some recent examples. Because of the specific aspects occurring in this scenario, however, we must consider a discussion of this type of investigations as outside of the scope of this article.

Since we do not restrict ourselves to some specific metaheuristic algorithm but would rather like to discuss the topic in a general context, we have to start with a generic framework encompassing most (or all) existing metaheuristics. This is done in Section 6.2. Section 6.3 addresses convergence definitions and different types of convergence of metaheuristic algorithms. In Section 6.4, core ideas of convergence proofs are outlined and illustrated by examples concerning different metaheuristics. Section 6.5 deals with convergence of metaheuristic variants for Stochastic CO problems, and Section 6.6 shortly discusses the issue of convergence speed. The final Section 6.7 contains concluding remarks and open research topics.

6.2 A Generic Metaheuristic Algorithm

We consider optimization problems of the form

$$f(x) \to \min \text{ such that } x \in S, \tag{6.1}$$

where S is a search space and f is a real-valued function called *objective function* or *cost function*. Maximization problems are reformulated as minimization problems in a straightforward way. We shall assume S to be a finite set, which puts the problem (6.1) into the field of CO.

To give a unified framework within which convergence may be discussed for a large class of metaheuristics, we present in the following a generic algorithm that includes most (perhaps all) currently used metaheuristic algorithms as special cases (cf. also [28]). In some sense, the proposed generic algorithm extends the generic black-box optimizer presented by Droste et al. [13], but it is not restricted to black-box optimization, and it does not consider prior function samples in their raw form, but rather in an already condensed form, which is closer to existing metaheuristics and automatically addresses space restrictions.

The algorithm (say, in a concrete instantiation A) works in an iterative way. In iteration t ($t \geq 1$), algorithm A uses a memory m_t and a list L_t of solutions $x_i \in S$. We call the elements of L_t *sample points*. The structure of the procedure is the following:

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 $g(m_t, \xi_t)$ of the memory state m_t and of a random influence ξ_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 state m_{t+1} as a function $h(m_t, L_t^+, \xi_t')$ of the current memory state m_t, of the list of solution-value pairs L_t^+, and of a random influence ξ_t'.

The memory state m_t can be decomposed as $m_t = (m_t^s, m_t^r)$ into two components m_t^s and m_t^r, where m_t^s, the *sample-generating part*, contains all information in m_t that is actually used by the function g for generating the list L_t of sample points, and m_t^r, the *reporting part*, contains that information that is not used for this purpose. The function h is allowed to use both parts m_t^s and m_t^r for updating the memory. Typically[1], the reporting part contains at least the *best-so-far* solution x_t^{bsf} which is initialized arbitrarily for $t = 1$, and set to x_i each time when in some iteration t, some $x_i \in L_t$ is evaluated to an objective function value $f(x_i)$ better than $f(x_t^{bsf})$. The best-so-far solution

[1] An exception is the context of optimization under noise discussed in Section 6.5, where the best-so-far solution cannot be determined with certainty.

is used as the currently proposed approximation to the optimal solution in iteration t.

The elements ξ_t and ξ'_t can be imagined as vectors of (pseudo-)random numbers that are used by the metaheuristic. Thus, the function $g(m_t, \xi_t)$ specifies, to given memory m_t, a probability distribution for the list of new sample points, whereas the function $h(m_t, L_t^+, \xi'_t)$ specifies, to given memory m_t and current list L_t^+ of solution-value pairs, a probability distribution for the new state of the memory. If the functions g and h are independent of ξ_t resp. ξ'_t, we obtain the special case of a *deterministic* search algorithm. Most metaheuristics, however, are stochastic search algorithms.

Contrary to [13], in our formalism, the functions g and h may use any information on the problem instance. (In order not to overload notation, the problem instance is not written explicitly as an additional argument of g and h.) 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 actual problem instance, is denoted as *black-box optimization*. Some variants of metaheuristics are black-box optimization algorithms, some are not. In particular, algorithms combining a metaheuristic with mathematical programming (see Section 6.1) are typical cases of algorithms that are *not* of black-box type.

The states (m_t, L_t^+) generated during the execution of the algorithm form a *Markov process* in discrete time, since the distribution of the next state (m_{t+1}, L_{t+1}^+) only depends on the current state (m_t, L_t^+). For fixed objective function f, we can consider already the sequence (m_t) $(t = 1, 2, \ldots)$ of memory states as a Markov process, since (via L_t^+, which results from m_t) the distribution of m_{t+1} only depends on m_t.

To illustrate that the generic algorithm above is able to cover well-known metaheuristics as special cases, let us look at some examples:

Generalized Hillclimbing Algorithms: This class of metaheuristics, which we shortly denote as GHCs (generalized hillclimbers), was introduced by Jacobson et al. [43] and contains the well-known *Simulated Annealing* (SA) algorithm and some other stochastic local search algorithms as a special cases. Here, a neighborhood structure $\mathcal{N}$ assigning to each $x \in S$ a set $\mathcal{N}(x) \subseteq S$ of *neighbor solutions* is used. The sample-generating part m_t^s of the memory consists of a single element, the current *search point* x_t. The reporting part m_t^r contains x_t^{bsf} as well as the current iteration counter t. The list L_t of sample points consists of a single element, the currently investigated neighbor solution $y \in \mathcal{N}(x)$ to x.

- To determine L_t from m_t^s, choose a random neighbor y to the element x in m_t^s.
- To update m_t to m_{t+1}, decide by the following stochastic acceptance rule whether y is accepted or not: Draw a random variable R from a distribution $\mathcal{D}_t$ depending on t. If $R \geq f(y) - f(x)$, accept y, i.e., set $m_{t+1}^s = \{y\}$. If

$R < f(y) - f(x)$, reject y, i.e., set $m_{t+1}^s = \{x\}$. The new reporting part m_{t+1}^r is obtained by updating x_t^{bsf} to x_{t+1}^{bsf} and by incrementing t.

SA is the special case where the random variable R is chosen as $-c_t \ln(\zeta)$, where c_t is the *temperature* parameter of SA in iteration t, and ζ is a random number distributed uniformly on $[0, 1]$. The acceptance probability of worse neighbors results then as $\exp(-(f(y) - f(x))/c_t)$.

In other stochastic local search algorithms such as Iterated Local Search (ILS) or the Variable Neighborhood Search (VNS) algorithm developed by Hansen and Mladenović [33], the memory is slightly extended, e.g., by the addition of an *incumbent* solution. Contrary to GHCs and to ILS, VNS works with a family of neighborhoods $\mathcal{N}_k$ of varying sizes $k = 1, \ldots, k_{max}$. In all these metaheuristics, the number of elements in the memory is fixed (and usually small) and is not tuned to the problem instance. This is different in the next example:

Genetic Algorithms (GAs): For the *canonical* GA according to the definition in [54], m_t^s consists of a current population of k solutions, m_t^r contains only x_t^{bsf}, and also L_t consists of k solutions.

- To determine L_t from m_t^s, apply the well-known genetic 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 by using the corresponding objective function values. The result is stored as m_{t+1}^s. The new reporting part is obtained by updating x_t^{bsf} to x_{t+1}^{bsf}.

Other GAs are represented as obvious modifications of this procedure.

Ant Colony Optimization (ACO): Both the *Ant System* (AS) variant by Dorigo et al. [10] and the *Max-Min Ant System* (MMAS) variant by Stützle and Hoos [58] can be represented in the following form: The sample-generating part m_t^s of the memory consists of a vector (or matrix) of real-valued parameters, called *pheromone* values. The reporting part m_t^r contains x_t^{bsf} and (depending on the special MMAS version) possibly also an *iteration-best* solution x_t^{ib}. The list L_t consists of k solutions.

- To determine L_t from m_t^s, let k "ants" construct k random solutions by traversing random paths in a "construction graph", where the probabilities of the moves are governed by the pheromone values in m_t^s.
- To update m_t to m_{t+1}, start by updating x_t^{bsf} and x_t^{ib} based on the k new solutions in L_t^+. Then, apply the specific *pheromone update rule* of the ACO variant under consideration in order to determine the new pheromone values from the current values by reinforcing the components of the solution(s) contained in x_t^{bsf} and/or x_t^{ib}. This gives m_{t+1}^s.

The macro-structure of *Estimation-of-Distribution Algorithms* (EDAs, see, e.g., Gonzalez et al. [20]) is very similar to that of ACO. The parameters of

the distribution used for sampling replace here the pheromone values of ACO. Also the *Cross Entropy Optimization* metaheuristic introduced by Rubinstein [53] shows the same macro-structure.

Finally, let us mention that also metaheuristics as *Particle Swarm Optimization* (PSO) (developed by Kennedy and Eberhart [47]) fit into the framework above, although in the standard version of PSO, search is not performed on a discrete finite search space, but on a continuous search space instead. PSO versions for CO problems as the Binary PSO algorithm proposed in [48] have a structure related to that just described for ACO. Here, m_t^s contains real-valued positions and velocities of a set of particles.

6.3 Convergence

6.3.1 Convergence Notions

The mathematical definition of convergence of a series $(x_1, x_2, \ldots)$ of elements in a space $\mathcal{X}$ with a distance function d is the following: The series (x_n) converges to a limit x^*, if for each $\epsilon > 0$, there is an integer N such that $d(x_n, x^*) < \epsilon$ for all $n \geq N$. This definition considerably simplifies if the space $\mathcal{X}$ is finite. In this case, x_n converges to x^* if and only if there is some N such that $x_n = x^*$ for all $n \geq N$. It should be mentioned, however, that although we restrict ourselves to finite *search spaces* in this paper, we shall also have to do with convergence of non-discrete random variables.

Most metaheuristics are *stochastic* search algorithms, such that the definition above is not sufficient. We need a generalization to a notion of convergence of a series of *random variables*. Let us start with a recapitulation of two important established definitions of *stochastic* convergence. After that, we proceed to a consideration of different types of convergence of a metaheuristic.

We consider a stochastic process $(X_1, X_2, \ldots)$, i.e., a sequence of random variables with a common distribution. In general, the random variables are not independent.

Definition 1.

(i) A sequence of random variables $(X_1, X_2, \ldots)$ converges *with probability one* (short: w. pr. 1) or *almost surely*[2] to a random variable X^*, if

$$Pr\{X_t \to X^*\} = 1, \tag{6.2}$$

[2] The last term is very usual in the probabilistic literature. We avoid it in this paper because the word "almost" can be misunderstood. If $X_t \to X$ w. pr. 1, it's quite sure that the sequence converges.

i.e., if with probability one, the realization $(x_1, x_2, \ldots)$ of the sequence (X_t) converges to the realization x^* of X^*.

(ii) A sequence of random variables $(X_1, X_2, \ldots)$ converges *in probability* to a random variable X^*, if for all $\epsilon > 0$,

$$Pr\{\, d(X_t, X^*) \geq \epsilon\} \;\rightarrow\; 0 \text{ as } t \rightarrow \infty, \tag{6.3}$$

where d is the distance function on the space $\mathcal{X}$ in which the random variables X_t take their values.

It can be shown that convergence notion (i) is stronger than convergence notion (ii): if $X_t \rightarrow X^*$ w. pr. 1, then also $X_t \rightarrow X^*$ in probability. In general, the converse does not hold. Let us construct, e.g., a sequence $X_1, X_2, \ldots$ of binary random variables as follows. Initialize each X_t by the value 0. Decompose the index set $\{1, 2, \ldots\}$ into blocks $k = 1, 2, \ldots$ of increasing length k, such that the first block is $\{1\}$, the second block is $\{2, 3\}$, the third block is $\{4, 5, 6\}$, etc. Now select within each block k a position t uniformly at random and set the corresponding variable X_t to the value 1. Then, $X_t \rightarrow 0$ in probability as $t \rightarrow \infty$, but $X_t \rightarrow 0$ w. pr. 1 does not hold, since a realization $(x_1, x_2, \ldots)$ of this stochastic process never converges.

Usually, these definitions are specialized to the case where the limiting X^* is a constant, deterministic element x^*. If $\mathcal{X}$ is a finite set, convergence of X_t to x^* w. pr. 1 holds exactly if

$$Pr\{\text{there is a } u \geq 1 \text{ such that } X_t = x^* \text{ for all } t \geq u\} \;=\; 1,$$

and convergence of X_t in probability holds exactly if $Pr\{\, X_t = x^*\} \rightarrow 1$ as $t \rightarrow \infty$. In the finite case, we can also slightly generalize the definition of convergence in probability by saying that X_t converges to a subset $\mathcal{X}^*$ of $\mathcal{X}$ in probability, if $Pr\{\, X_t \in \mathcal{X}^*\} \rightarrow 1$ as $t \rightarrow \infty$.

We shall apply these definitions to components of the current state (m_t, L_t^+) of the Markov process associated with algorithm A. However, which components should be considered, and how is the limiting element supposed to look like?

6.3.2 Best-So-Far Convergence

A natural choice consists in considering the best-so-far solution x_t^{bsf} and to ask whether or not it converges to some *optimal solution*, as $t \rightarrow \infty$. In a finite search space and with x_t^{bsf} defined as in Section 6.2, convergence of x_t^{bsf} to an optimal solution amounts to convergence of the cost function values $f(x_t^{bsf})$ $(t = 1, 2, \ldots)$ to the optimal cost function value. Thus, we may ask under which conditions it can be guaranteed that $f(x_t^{bsf})$ converges ("w. pr. 1" or "in probability") to $f^* = \min\{f(x) : x \in S\}$.

Restricting the discussion to the case of finite S, we can simplify the defining conditions (6.2) – (6.3) for convergence of $f(x_t^{bsf})$ to the optimum f^* by introducing the nondecreasing sequence of indicator variables

$$Z_t = I(f(x_t^{bsf}) = f^*) \quad (t = 1, 2, \ldots),$$

where I denotes the indicator function. The *success indicator* Z_t is 1 if an optimal solution is found in one of the iterations $1, \ldots, t$, and 0 otherwise. The *first hitting time* is given as

$$T_1 = \min\{t \geq 1 : \ Z_t = 1\}. \tag{6.4}$$

Furthermore, with E denoting the mathematical expectation, we define

$$\mu_t = E(Z_t) = Pr\{Z_t = 1\} = Pr\{T_1 \leq t\} \quad (t = 1, 2, \ldots) \tag{6.5}$$

as the probability that algorithm A finds an optimal solution in one of the iterations $1, \ldots, t$. (In some articles such as [35] or [64], the sequence of numbers $1 - \mu_t$ is called the *convergence rate*.) It is easy to see that with this notation and with the ordinary absolute difference on the set of reals as the distance function,

- $f(x_t^{bsf})$ converges to f^* w. pr. 1 if and only if $Pr\{Z_t = 0 \text{ for all } t\} = 0$ (which is the same as $Pr\{T_1 < \infty\} = 1$), and
- $f(x_t^{bsf})$ converges to f^* in probability if and only if $\mu_t \to 1$ $(t \to \infty)$.

It follows that for best-so-far convergence, the two convergence notions coincide, since $Pr\{Z_t = 0 \text{ for all } t\} \leq Pr\{Z_u = 0\} = 1 - \mu_u$ for all iterations u, such that $\mu_u \to 1$ as $u \to \infty$ implies convergence w. pr. 1.

The convergence concept above may look nice. However, it has a serious disadvantage: It turns out that under this concept, even very inefficient search algorithms converge to the optimum, which makes the concept too "generous". The standard example for this observation is random search.

In our framework, (pure) random search can be described as that instantiation of our generic algorithm where the sample-generating part m_t^s of the memory is empty, the reporting part m_t^r consists only of x_t^{bsf}, and the list L_t consists of a single sample point x_t that is chosen at random from S according to some fixed distribution. Because m_t^s does not contain any information, the choice of x_t has to be performed *independently* of the memory state m_t (and hence of the previous iterations).

It is well-known that random search on a finite set S ensures convergence of the best-so-far solution value to the optimum, as long as every solution $x \in S$ has a nonzero probability $p(x) > 0$ of being chosen as the sample point x_t. We will derive this quickly in Subsection 6.4.1. Furthermore, however, it will be shown that the expected value $E(T_1)$ of the first hitting time is usually very large for random search, such that convergence is here of no help for practice.

Informally, the reason why convergence in the best-so-far sense does not go hand in hand with a good runtime behavior is that for demonstrating this type of convergence, it is only required that the algorithm performs a sufficient amount of *exploration* of the search space. The feature of *exploitation* of the information obtained in previous iterations (stored in the memory m_t), as it is incorporated in most well-performing metaheuristics, does not alleviate the convergence proof; quite contrary, it rather hampers it. Of course, however, exploitation is an advantage for the runtime of the algorithm! Thus, convergence of the best-so-far solution value is not an indicator for a good runtime behavior. It only ensures that the part of the search space containing the optimal solution is not excluded from the search *a priori*, such that at least in the absence of limits on computation time, search is "complete".[3]

Several investigations on the convergence of diverse metaheuristics have started with results concerning the best-so-far concept. Let us give examples from two metaheuristic fields: Hartl [34] and Rudolph [54] showed convergence of $f(x_t^{bsf})$ to the optimum for certain variants of GAs transferring "elite" solutions (best solutions in a current population) from generation to generation. If, e.g., one elite solution is always preserved within the population, this solution is identical to x_t^{bsf}. Brimberg et al. [7] showed convergence results of best-so-far type for ILS as well as for a VNS parametrization where the parameter k_{max} defining the largest neighborhood is sufficiently large to cover the whole search space S. The proofs provided in the mentioned articles are possible starting points for the derivation of stronger convergence properties (cf. the remarks in the next subsection).

6.3.3 Model Convergence

The chance that provable convergence properties are correlated with good runtime behavior are considerably increased if we do not focus on the best-so-far solution x_t^{bsf}, but on the sample-generating part m_t^s of the memory. In an efficient metaheuristic, exploitation of the search experience should concentrate the search more and more on the most promising areas of the search space S, with the consequence that the average cost function values of the sample points in L_t tend to decrease (not necessarily monotonically) over time. The case where the expected cost function values in L_t remain constant is essentially the exploitation-less random search case.

Borrowing from the concept of "model-based search" as developed by Zlochin et al. [65], we may alternatively denote the sample-generating part m_t^s of the memory as the current *model* for the search distribution. In the model-based view, search points are generated in dependence of the model,

[3] Hoos [39] and Hoos and Stützle [40] call a search algorithm with $Pr\{T_1 < \infty\} < 1$ for a class of problems *essentially incomplete* for this class.

cost function values are evaluated, and the obtained information is then fed back into a modification of the model. Basically, this corresponds to the mechanism of our generic algorithm, with the difference that we also extend this view to classical search algorithms with discrete state spaces instead of restricting it to metaheuristics as ACO, EDAs or Cross Entropy Optimization, where the model is described by a vector of *real-valued* parameters.

By the argumentation above, a runtime behavior superior to that of random search can be expected if it can be shown that the *model* m_t^s converges, as $t \to \infty$, to some limiting state $(m^s)^*$ that supports only the generation of optimal or at least high quality sample points. We denote this type of convergence as *model convergence* as opposed to best-so-far convergence. Note that the model can be very small: in a GHC, e.g., it contains only the current search point x_t. Nevertheless, convergence of this search point to a solution in S^* is more relevant than convergence of x_t^{bsf} resp. $f(x_t^{bsf})$ only, since it indicates that the search is gradually directed towards more promising areas of the search space.[4]

Contrary to proofs of best-so-far convergence which are technically the easier the more emphasis the considered algorithm puts on exploration (as opposed to exploitation), model convergence proofs have to take the *exploration-exploitation tradeoff* explicitly into account and only succeed under parameter assumptions ensuring a proper balance between these two factors. Typically, the results yield rather narrow conditions for parameter schemes within which model convergence holds; outside the balanced regime, either a surplus of exploitation yields premature convergence to a suboptimal solution, or a surplus of exploration produces random-search-type behavior without model convergence (although best-so-far convergence may hold).

Historically, the first model convergence results have been found in the SA field (see, e.g., Gelfand and Mitter [17], Hajek [32], or Aarts and Korst [1]). In Subsection 6.4.2, we shall outline the key ideas of the proofs in the more general context of modern GHC results. Also for some ACO variants and for a variant of Cross Entropy Optimization, results of model convergence type are known. Empirical evidence suggests that such results should be possible for several other metaheuristics as well.

In the GA case, model convergence would mean that the population tends to "positive stagnation" in the long run by being filled with optimal solutions only. Since mutation counteracts this effect, a model-convergent variant of a GA would presumably have to gradually decrease the mutation rate and/or to increase the selection pressure. This would have to be done slowly enough

[4] Convergence analysis of metaheuristics has been questioned by the argument that every randomized search algorithm can easily be made convergent to the optimum by repeatedly calling it with randomly chosen start solutions. This is trivial for best-so-far convergence, as already the series of start solutions yields a random search run in itself. However, *model* convergence typically does *not* hold in this scenario, since during each restart, the current information in m_t^s is thrown away. Thus, the expected average cost function value in the sample points does not improve from run to run.

to prevent premature convergence and fast enough to ensure convergence. Similarly investigation could be performed for the VNS case. (Convergence of the Markov process of VNS visits in local optima has been shown in Brimberg et al. [7].)

6.4 Proving Convergence

6.4.1 Proving Best-So-Far Convergence

Convergence of x_t^{bsf} to the optimum can usually be shown easily. We shall illustrate this for the simple case of random search. Let Z_t be defined as in Section 6.3, and let $p = \sum_{x \in S^*} p(x) > 0$ denote the probability of hitting an optimal solution in a single iteration of the random search algorithm. Because of the independence of the trials in different iterations, $Pr\{Z_t = 0\} = (1-p)^t \to 0$ as $t \to \infty$, and therefore $\mu_t \to 1$ as $t \to \infty$. In other words, convergence in probability to the optimum holds (and by the remark in Subsection 6.3.2 on the special situation for best-so-far convergence, even convergence w. pr. 1).

In some more interesting algorithms, the hitting probability is time-dependent: $p = p_t$. If a strictly positive lower bound $p_{min} > 0$ for p_t can be shown, convergence immediately results as above. However, even in some cases where p_t tends to 0, convergence still holds: E.g., if $p_t = ct^{-1/2}$ $(c > 0)$, we still get convergence in probability since $\lim_{t\to\infty}(1 - ct^{-1/2})^t = 0$. For $p_t = c/t$, this does not hold anymore, since $(1 - c/t)^t \to e^{-c} > 0$. Lower bounds on the hitting probability have been used, e.g., in the convergence analysis of the MMAS variant of ACO given by Stützle and Dorigo [57].

For time-independent p, the expected value $E(T_1)$ of the first hitting time computes as $1/p$ by the standard calculation of the expected value of a geometric distribution, which is typically a very large number even for medium-sized search spaces, unless if the distribution $p(\cdot)$ of the random trials can be focused around the set S^* of optimal solutions by means of prior information.

6.4.2 Proving Model Convergence

We outline the ideas of model convergence proofs by presenting some characteristic examples.

6.4.2.1 Generalized Hillclimbers and Simulated Annealing

The results by Jacobson and Yücesan [44] concerning convergence of GHCs are especially instructive as, on the one hand, GHCs contain SA (for which the first model convergence theorems have been shown) as a special case, and on the other hand, the article works already on a more general level such that metaheuristic-independent features become visible. In [44], it is assumed that subsequent iterations of a GHC are comprised to *macro iterations* $k = 1, 2, \ldots$. (We shall choose a simple, special way of defining macro iterations later.) During each macro iteration, the random variable R deciding on acceptance (see Section 6.2) has the same distribution. With $t(k)$ denoting the last "micro" iteration of macro iteration k, let x^k denote $x_{t(k)}$, i.e., the current search point as it is obtained at the end of macro iteration k. We introduce the following abbreviations:

- $C(k)$ is the event that $x^k \in S^*$, i.e., the event that macro iteration k produces an optimal solution. The complementary event is denoted by $C^c(k)$.
- $B(k)$ is the event $C^c(1) \cap C^c(2) \cap \ldots \cap C^c(k)$, i.e., the event that none of the macro iterations $1, \ldots, k$ produces an optimal solution. The complementary event to $B(k)$ is $B^c(k)$.
- $B = \bigcap_{k=1}^{\infty} B(k)$ is the event that no iteration at all produces an optimal solution.
- $r(k) = Pr\{B^c(k) \,|\, B(k-1)\}$ is the probability that in macro iteration k, an optimal solution is produced, although it has not yet been produced in any of the previous macro iterations.

Convergence of x^k in probability to the set X^* of optimal solutions can be expressed as $Pr(C(k)) \to 1$ as $k \to \infty$. It is now possible to show the following criterion:

Theorem 1 (Jacobson and Yücesan [44]). For a GHC, x^* converges to S^* in probability if and only if the two following two conditions are satisfied:

(i) $\sum_{k=1}^{\infty} r(k) = \infty$,
(ii) $\Pr(C^c(k) \,|\, B^c(k-1)\} \to 0$ as $k \to \infty$

Let us present the proof idea of the part of the theorem stating that the two conditions above are *sufficient* for convergence in probability. The idea is not too complicated, but very informative, because it recurs in some variations in related proofs in the literature. First, we show that condition (i) is equivalent to $Pr(B) = 0$. This results as follows:

$$Pr(B) = Pr(B(1)) \cdot Pr(B(2)|B(1)) \cdot Pr(B(3)|B(1) \cap B(2)) \cdot \ldots$$

$$= (1 - r(1)) \cdot (1 - r(2)) \cdot (1 - r(3)) \cdot \ldots .$$

Therefore,

$$Pr(B) = 0 \Leftrightarrow \prod_{k=1}^{\infty}(1 - r(k)) = 0 \Leftrightarrow \sum_{k=1}^{\infty}\log(1 - r(k)) = -\infty,$$

where the second equivalence follows by taking logarithm on both sides. Since $\log(1 - r(k)) \sim -r(k)$ for small $r(k)$, the latter is equivalent to $\sum_{k=1}^{\infty} r(k) = \infty$. (To make the proof precise, the approximation has to be replaced by bounds.)

Now, by the law of total probability,

$$\begin{aligned} Pr(C^c(k)) &= Pr(C^c(k)|B^c(k-1)) \cdot Pr(B^c(k-1)) \\ &\quad + Pr(C^c(k)|B(k-1)) \cdot Pr(B(k-1)) \\ &= Pr(C^c(k)|B^c(k-1)) \cdot Pr(B^c(k-1)) + P(B(k)). \end{aligned}$$

By condition (ii), the first term in the last expression tends to 0 as $k \to \infty$. Because of the equivalence derived above, condition (i) yields $Pr(B) = 0$. Because $B(1) \subseteq B(2) \subseteq \ldots$, by the Monotone Convergence Theorem,

$$Pr(B(k)) \to Pr\left(\bigcap_{k=1}^{\infty} B(k)\right) = Pr(B) = 0,$$

and therefore also the second term tends to zero. This shows $Pr(C^c(k)) \to 0$, which completes the proof.

Theorem 1 can be nicely interpreted in terms of the exploration-exploitation tradeoff: Condition (i) guarantees that enough exploration is performed in order to be sure to find a globally optimal solution eventually. Condition (ii), on the other hand, ensures that enough exploitation is done in order to preserve an optimal solution with a high probability, once it has been found, and enables convergence in this way.[5]

Consider now the special case of SA. We choose macro iterations of equal length, consisting of L micro iterations each, where L is the maximum of the minimum number of transitions to neighbors required to reach an optimal solution from an arbitrary initial solution x over all $x \in S$. Then it is possible to reach from an arbitrary point $x \in S$ an optimal solution in exactly L micro

[5] The decomposition of the process into a phase before and a phase after an optimal solution x^* has been found may appear as a cheap trick: One might be tempted to construct a "model-convergent" metaheuristic by letting it perform random search before x^* has been found, and to freeze the current solutions x_t to x^* after that time. The point is, however, that the decomposition into these two phases only exists at the level of analysis and cannot be done by the algorithm itself, which does not know when it has detected an optimal solution, such that it cannot use the attainment of the optimum as the criterion for switching from exploration to exploitation. Thus, in order to preserve x^* after it has been discovered, the algorithm has to do a sufficient amount of exploitation already before this event — which, on the other hand, makes it nontrivial to guarantee that the global optimum is not missed.

iterations (i.e., in one macro iteration), either by moving with the search point x_t towards x^* or by letting it stay in x^*, rejecting neighbor solutions. We shall focus on the process (x^k) defined by the macro iterations, but mention that corresponding results can also be derived for the process (x_t) on the level of micro iterations.

It is easy to see that if the temperature parameter is fixed at some constant level c, either condition (i) or condition (ii) of Theorem 1 are violated: If $c > 0$, then $r(k)$ has a strictly positive lower bound, such that condition (i) holds; in this case, however, condition (ii) is not satisfied, since even after an optimal solution has been visited, suboptimal solutions will always be produced with probabilities larger than some positive constant. On the other hand, if $c = 0$, then only better neighbor solutions are accepted, with the consequence that condition (ii) is satisfied (once an optimal solution has been found, it is not left anymore), but condition (i) is violated, because usually an optimal solution is not found.

The stunt of satisfying the two conditions simultaneously is achieved by decreasing the temperature parameter with a suitable speed. Choose a temperature scheme c_k with $c_k \to 0$ $(k \to \infty)$ and $c_k \geq L\Delta/\log(k+1)$ $(k = 1, 2, \ldots)$, where $\Delta = \max_{x,y \in S}(f(x) - f(y))$. It is easily seen that as soon as the temperature has become low enough,

$$r(k) \geq \left[\frac{1}{|S|} \cdot \exp\left(-\frac{\Delta}{c_k}\right)\right]^L \geq \frac{C}{k}$$

with some constant $C > 0$, which implies that $\sum r(k) = \infty$ and hence condition (i) of Theorem 1 is satisfied.

To show that the above temperature scheme is also sufficient for satisfying condition (ii) needs some technicalities from the theory of inhomogeneous Markov chains, which we omit here; the interested reader is referred to [1]. Let us only provide the rough picture. The sequence (x_t) is an inhomogeneous Markov chain with transition matrix $P(k)$ on temperature level c_k. Condition (i) above ensures *weak ergodicity* of this Markov chain, which essentially means that the dependence on the initial solution vanishes over time. To satisfy also condition (ii), it has to be shown that (a) for all k, there exists a left eigenvector $\pi(k)$ of $P(k)$ with eigenvalue 1, (b) the eigenvectors $\pi(k)$ satisfy $\sum_{k=1}^{\infty} ||\pi(k) - \pi(k+1)|| < \infty$, and (c) the eigenvectors π_k converge as $k \to \infty$ to a limiting vector π^* containing probabilities of the solutions $x \in S$ such that only the probabilities in S^* have nonzero values. These properties can be demonstrated to be satisfied indeed for the given temperature scheme, which implies that the Markov chain is strongly ergodic and converges in distribution to π^*.

6.4.2.2 Ant Colony Optimization and Cross Entropy Optimization

The convergence proofs for special ACO variants in [21, 22] and for a particular variant of Cross Entropy Optimization in [49] have a similar structure as the results for GHCs outlined above. Let us explain this for the ACO case.

The algorithmic variants investigated in [22] are derived from the MMAS variant of ACO proposed in [58]. MMAS provides the possibility of applying a *lower pheromone bound* $\tau^{min} > 0$ which prevents that the components of the pheromone vector τ_t contained in m_t^s approach zero. In this way, exploitation is limited to a certain degree in favor of exploration. The update of the pheromone values is done by a reinforcement of the components of the best-so-far solution x_t^{bsf}, sometimes also the components of the iteration-best solution x_t^{ib} are reinforced. The degree of reinforcement is controlled by a parameter $\rho \in]0, 1[$ called *evaporation rate*, which can be considered as a learning rate. The new pheromone vector results as

$$\tau_{t+1} = (1 - \rho)\tau_t + \rho\psi_t,$$

where ψ_t is the vector of the rewards in the current iteration. If ρ is high, the observations made in the current iteration (the "presence") have a high influence on the new pheromone vector, compared to former iterations (the "past"), whereas in the more conservative case of low ρ, the past is given a higher influence than the presence.

In [22], two particular variants are considered: The first of them, "algorithm 1", does not use a pheromone bound, but decreases the learning rate ρ over time, choosing it as $\rho = \rho_t$. The second one, "algorithm 2", uses a time-dependent lower pheromone bound τ_t^{min}. In both algorithms, iteration-best reinforcement is not done, i.e., x_t^{ib} is not used.

The following result is shown: Both for algorithm 1 and for algorithm 2, as $t \to \infty$, x_t^{bsf} converges w. pr. 1 to an optimal solution x^*, *and* the current state τ_t of the sample-generating part m_t^s of the memory converges w. pr. 1 to a pheromone vector τ^* allowing only the generation of the optimal solution x^*, provided that the following conditions are satisfied:

- In the case of algorithm 1:

$$\rho_t \leq 1 - \frac{\log t}{\log(t+1)} \quad \text{and} \quad \sum_{t=1}^{\infty} \rho_t = \infty.$$

  This can be achieved, e.g., by the parameter scheme $\rho_t = c/(t \log t)$ with $0 < c < 1$.
- In the case of algorithm 2:

$$\tau_t^{min} = c_t / \log(t+1) \quad \text{with} \quad \lim_{t\to\infty} c_t > 0.$$

  This can be achieved, e.g., by constant $c_t = c > 0$.

The line of the proof is similar to the general scheme implicit in the proof of Theorem 1. Also here, the first part of the proof consists in ensuring that eventually, an optimal solution x^* is found. For this purpose, it is demonstrated that from the indicated conditions, a counterpart to condition (i) in Theorem 1 follows. After that, it has to be ensured that w. pr. 1, pheromone concentrates on the components of x^* and vanishes elsewhere. (That the optimal solution x^* is not left anymore after the first visit is trivial here by construction of x_t^{bsf}.) The convergence of the pheromone vector results by a deterministic consideration for an arbitrary realization of the stochastic process, such that we even obtain convergence w. pr. 1 in this scenario. Note that the conditions for algorithm 1 require that ρ_t decreases neither too fast nor too slow, and the conditions for algorithm 2 require an analogous property for the lower pheromone bounds τ_t^{min}.

Whereas in [22] it was assumed that the rewards for the components of the best-so-far solution are of a constant amount, Sebastiani and Torrisi [55] gave convergence conditions for the modification of the MMAS algorithm where the size of the rewards is chosen "fitness-proportional", i.e., decreasing in the cost function value of the solution to be reinforced. (This modification is frequently used in practice.)

Margolin [49] was able to provide a convergence proof for a certain Cross Entropy Optimization variant. Both convergence conditions and proof technique are closely related to the results in [22], so that we omit the details here.

6.4.2.3 Practical Aspects

What do the outlined results imply for applications of metaheuristics? We do not claim that it is always advisable to use a model-convergent parameter scheme when implementing a metaheuristic. Rather than that, our claim is that it can be helpful to know how such a scheme looks like. Let us justify this by an informal argument: In some sense, a model-convergent parameter scheme is a scheme maximizing exploitation under the constraint that there is still a sufficient amount of exploration in order to keep the promise of finally achieving the globally optimal solution alive. In cases of large problem instances with many good local optima, it may be more efficient to sacrifice the warranty of finding the global optimum "at the end of the day" for the benefit of accelerating the search for good local optima "in the course of the day". In such a case, one may decide to apply a parametrization of the algorithm that is slightly below the model-convergent scheme with respect to the degree of exploration. This means slightly faster cooling in SA or slightly less restrictive lower pheromone bounds in ACO, compared to the conditions in the theoretical convergence results.

Future research, both on a theoretical and on an experimental level, may possibly concretize this consideration by quantitative results.

6.5 Convergence for Problems with Noise

There is a situation where the concept of best-so-far convergence is not applicable at all. It is the scenario of *Stochastic Combinatorial Optimization* characterized by the property that the parameters of either cost function or constraints are not known with certainty, such that stochastic models are required to represent these parameters as random variables. We restrict ourselves to the special (but frequently occurring) case where only the cost function is uncertain, and the objective is to minimize its expected value. If this expected value can be computed efficiently, the situation reduces to that of deterministic CO and can be treated by ordinary metaheuristics. Otherwise, a solution algorithm has to be based on concrete observations (realizations) of the cost function values in certain points $x \in S$, but these observations do not represent the expected value, but deviate from it by "noise". A typical case is that where expected costs are estimated by (Monte Carlo) simulation, as it is frequently done, e.g., in queuing systems or in stochastic vehicle routing.

The SCO problem has then the following general form:

$$E(f(x,\omega)) \rightarrow \min \text{ such that } x \in S. \tag{6.6}$$

Therein, ω denotes a random influence (with a distribution given by the stochastic model of the problem), which has to be distinguished from the random variables ξ and ξ' used by the metaheuristic solution procedure (see Section 6.2), and E is the expectation with respect to the distribution of ω. For surveys on the solution of problems of this type by metaheuristics, see Jin and Branke [46] and Bianchi et al. [3].

A *sample average estimator* (SAE) approximates the true objective function value $F(x) = E(f(x,\omega))$ by the average over a random sample:

$$\tilde{F}(x) \;=\; (1/s) \sum_{\nu=1}^{s} f(x,\omega_\nu) \;\approx E(f(x,\omega)). \tag{6.7}$$

Therein, ω_ν $(\nu = 1,\ldots,s)$ represent s independent sample observations, e.g., s runs of a simulation routine. It is immediately seem that $E(\tilde{F}(x)) = F(x)$, i.e., the SAE is always unbiased. We can apply the SAE during a metaheuristic optimization run every time when an objective function evaluation is required, but we have to keep in mind that $\tilde{F}(x)$ usually deviates from $F(x)$, and this is so even more if the sample size s is small. Increasing the sample size s improves the accuracy of the SAE, but this comes at the price of increasing the runtime. The resulting tradeoff has to be addressed by efficient metaheuristic variants.

Several modifications of different metaheuristics have been proposed in the literature for the treatment of SCO problems, e.g., in the SA field [18, 30, 2], in the ACO field [23, 24, 4] or in the VNS field [29]. These techniques are usually

variable-sample approaches extending the generic algorithm of Section 6.2 in the following way: In step 2b of the generic algorithm, the evaluation of the objective function values $f(x_i)$ is replaced by the determination of an SAE $\tilde{F}(x_i)$. Typically, the sample size $s = s_t$ is increased from iteration to iteration, but there are also variants where s is kept constant.

A crucial observation is that in this scenario, the best-so-far solution x_t^{bsf} cannot be determined anymore, since we cannot decide with certainty which of two solutions x and y has the better objective function value; by increasing s, a guess based on the SAEs $\tilde{F}(x)$ and $\tilde{F}(y)$ becomes more trustable, but never certain. Convergence issues get very important in this framework, because in a non-convergent situation, we would neither know which of the visited solutions to deliver as the proposed solution nor when to stop the algorithm: diminishing marginal gains of $f(x_t^{bsf})$, as they can be used as a stopping criterion in the deterministic context, do not give us a hint here.

The key idea to obtain convergence in the outlined stochastic context is to carefully control the accuracy of the estimates $\tilde{F}(x) = \tilde{F}_t(x)$ as t increases by corresponding increments of the sample size $s = s_t$ in iteration t, such that the influence of randomness is gradually reduced while the metaheuristic under consideration turns to more promising areas of the search space. As a consequence, it becomes more probable in later phases that the global optimum is recognized as such by the search procedure, whereas in earlier phases, where the average solution quality is only small, it would be a waste of time to strive for a very good accuracy in objective function evaluation.

In [30], this idea is carried out by proving that, on some conditions, convergence in probability of SA carries over to the stochastic context. The most essential condition is that the standard deviations of the noise variables $\tilde{F}_t(x) - F(x)$ decrease as $t \to \infty$ with an order $O(t^{-\gamma})$ where $\gamma > 1$. Since the variance of the SAE is inversely proportional to the sample size, this can be achieved by letting the sample size s_t grow faster than quadratically in t.

A related approach has been followed in [23] for proving convergence of a variant S-ACO of ACO proposed for SCO problems. In the design of S-ACO, an attempt has been made to improve performance by considering, instead of the current search point x_t, something that corresponds to the best-so-far solution x_t^{bsf} in the deterministic context: Let $\hat{x}_t^{bsf}$ denote the *presumably* best-so-far solution in iteration t, defined by an arbitrary initialization in iteration 1 and a replacement of $\hat{x}_t^{bsf}$ each time when in some subsequent iteration, the SAE $\tilde{F}_t(x_t)$ of the current solution x_t found in this iteration turns out as better than the SAE $\tilde{F}_t(\hat{x}_t^{bsf})$. In other words, at the end of each iteration t, we perform a *tournament* between the current $\hat{x}_t^{bsf}$ and the current x_t, based on a sample of size s_t, and define the new presumably best-so-far solution as the winner of this tournament. It is shown in [23] that for a modification of the MMAS "algorithm 2" described above with a sample size s_t growing at least linearly in t, convergence of $\hat{x}_t^{bsf}$ to the optimal solution of (6.6) is ensured.

The tournament concept has also been used in [29] to present a provably convergent version S-VNS of VNS for SCO problems. In the proof, a general theorem by Homem-de-Mello [8] is applied, which can possibly also be useful in the convergence analysis of other SCO-metaheuristics. Therefore, let us shortly outline its idea.

Denote the vector of independent random numbers that are used in the tournament of iteration t for the evaluation of $\tilde{F}(\hat{x}_t^{bsf})$ and of $\tilde{F}(x_t)$ (we can take the same vector for both SAEs) by $\omega^t = (\omega_1^t, \ldots, \omega_{s_t}^t)$. The same solution x can take part in the tournament in several iterations, possibly even infinitely often. An interesting question is how fast the sample sizes s_t have to be increased such that we can be sure that after sufficient time, a suboptimal x is distinguished reliably from an optimal x, and (if present) only optimal solutions will win the tournament in the future. Homem de Mellos's theorem, which is proved by large-deviations theory, answers this question:

Theorem 2 (Homem-de-Mello [8], Proposition 3.2). Suppose that for a scheme $(s_1, s_2, \ldots)$ of sample sizes and independent random variables ω_ν^t,

(i) for each $x \in S$, the variances $var[f(x, \omega_1^t)]$ are bounded by some constant $M(x) > 0$,

(ii) the variables ω_ν^t are identically distributed, and the SAEs

$$\tilde{F}_t(x) = (1/s_t) \sum_{\nu=1}^{s_t} f(x, \omega_\nu^t)$$

are unbiased[6], i.e., $E(\tilde{F}_t(x)) = F(x)$ for all x,

(iii) $\sum_{t=1}^{\infty} \alpha^{s_t} < \infty$ for all $\alpha \in]0, 1[$.

Then for each x, we have $\tilde{F}_t(x) \to F(x)$ $(t \to \infty)$ w. pr. 1.

In the S-VNS algorithm [29], a tournament is only performed at the end of each macro iteration, where a macro iteration consists of a shaking step followed by local search. For simplicity, let us apply t as an index for macro iterations in the following. Convergence of S-VNS is shown as follows: First of all, it is demonstrated that with a probability larger than zero, a macro iteration finds an optimal solution x^* and exposes it to the tournament. With probability one, this even happens in infinitely many macro iterations. By using Theorem 2, it is verified that for a specific realization of the stochastic process, among all macro iterations where x^* is exposed to the tournament, there is one (say, macro iteration t^*) from which on the sampling error is already small enough to distinguish reliably between optimal and suboptimal solutions. In t^* and in all subsequent macro iterations, an optimal solution will win the tournament, which proves the assertion.

[6] [8] also refers to a more general situation where $E(f(x, \omega_\nu^t))$ can be different from $F(x)$. In our context, unbiasedness follows by definition.

Condition (iii) of Theorem 1 requires that s_t grows fast enough. It is easy to see that a growth proportional to $t^{1/2}$ already satisfies condition (iii), whereas a growth proportional to $\log t$ is yet too slow. The other conditions of Theorem 2 are usually automatically satisfied.

We see that the proof relies on a convergence property of (only) best-so-far type for the underlying *deterministic* VNS algorithm, which makes the result weaker than those for SA and for ACO. It would be desirable to extend it to a result of model-convergence type.

6.6 Convergence Speed

After having ensured convergence of a metaheuristic algorithm A, the natural next question is "How fast does A converge on a given problem instance?"[7] We have seen in the previous sections that for several metaheuristics, very general convergence results, covering the whole range of CO problems, can be derived. Unfortunately, it is rather unlikely that the next step can be to show comparably broad positive results for the speed of convergence. The reason lies in the so-called *No-Free-Lunch Theorems* (NFLTs) by Wolpert and Macready [63] stating that in the average over *all* cost functions $f : S \to W$, where W is some value range, the expected performance of every black-box optimization algorithm is the same. In particular, in this average, no metaheuristic A is better than random search, and to every function f_1 where A outperforms random search, there must be another function f_2 where random search outperforms A.

The chance to prove a convergence speed faster than that of random search for an algorithm increases for restricted sets of problems or problem instances. NFLTs do not hold within problem classes with restricted computational complexity (see [11, 14, 15, 5]) or for objective functions with certain fitness landscape properties (see [41, 42]). At the moment, however, it has not yet been achieved to derive *general* convergence speed bounds using these observations. Rather than that, the literature on the optimization time of metaheuristic algorithms is split into a large number of single results for special algorithms applied to special (usually rather simple) objective functions. It cannot be the goal of this paper to survey these results. Overviews have recently been given in [51] for evolutionary algorithms with the exception of the swarm-intelligence algorithms ACO and PSO, and in [26] for ACO. However,

[7] It does not make too much sense to raise the second question before the first is answered at least for special problem instances and in the weakest possible meaning. E.g., if for an algorithm A on a given problem instance, x_t^{bsf} does not converge in probability to a set $\tilde{S}$ of solutions considered as sufficiently good (which can be the set S^*), then the expected first hitting time of $\tilde{S}$ is ∞. The concept of convergence or runtime "with overwhelming probability" has been used to deal with situations where non-convergence cannot be excluded, but its interpretations are distinctly less intuitive than those of expected runtimes.

a short recapitulation of some main tools for analyzing convergence speed applicable to *several* metaheuristics may be helpful.

For the sake of brevity, we focus on the possibly most important performance measure, the expected first hitting time $E(T_1)$ of the optimal solution, where T_1 is defined by (6.4). Note that in view of (6.5), the distribution function of T_1 is given by $t \mapsto \mu_t$. If, from the viewpoint of application, it is sufficient to reach a solution of some fixed minimum quality instead of the exact optimum, the concept can easily be modified by considering $E(\tilde{T}_1)$ instead of $E(T_1)$, where $\tilde{T}_1$ is given as $\min\{t \geq 1 : \ f(x_t^{bsf}) \leq c\}$ with some aspiration level c for the cost function.

The following outline of methods is neither intended to be complete nor to present all the formal details. Rather than that, the aim is the presentation of some often applied basic ideas.

(1) Markov Chain Analysis

In the case where the memory content m_t can only take finitely many values, the property that the stochastic process (m_t) is a Markov process can be used directly for computing $E(T_1)$ — at least in principle. In this case, (m_t) is a (homogeneous) *Markov chain*. Examples are GHCs (including SA) and GAs. For the ease of notation, let us assume in the sequel that the optimal solution x^* is unique, and that even the state m^* of the memory producing the optimal solution is unique; this scenario can easily be generalized to the existence of several optimal memory states. We ask for the expected time until state m^* is visited first. Let $P = (p_{ij})$ $(1 \leq i, j \leq N)$ denote the transition matrix of the Markov chain. Arrange the indices of the possible memory states in such a way that the largest index N corresponds to m^*, and let $\hat{P}$ denote the matrix obtained from P by deleting the Nth column and the Nth line. Then it is well-known (see, e.g., Mühlenbein and Zimmermann [50] or He and Yao [37]) that the vector $\vartheta = (\vartheta_1, \ldots, \vartheta_{N-1})$ of the values for $E(T_1)$ after a start in memory state $1, \ldots, N-1$, respectively, computes as

$$\vartheta = (I - \hat{P})^{-1} \cdot 1_{N-1}, \tag{6.8}$$

where I is the identity matrix, and 1_{N-1} is the vector consisting of $N-1$ ones. The inversion of the matrix $I - \hat{P}$ may cause difficulties, but in some special cases, explicit formulas can be derived. E.g., [50] gives a formula for the case where $I - \hat{P}$ has a tri-diagonal form, i.e., contains only three nonzero elements in each line.

(2) State Space Decomposition and Subgoals

Usually, the state space becomes to large to be tractable directly by (6.8). However, in some cases, it can be decomposed into subsets that are treated equivalently by the metaheuristic under consideration. If the transition probability between an element of subset i and an element of subset j only depends on i and j, then the Markov property is also satisfied for the "embedded" chain which has the subsets as its states. This can reduce the size of the state

space considerably. To give a simple concrete example, let us consider the function

$$\sigma_{\alpha,\beta,n}(y) = \begin{cases} y, & \text{if } 0 \le y < \alpha, \\ -y + 2\alpha, & \text{if } \alpha \le y < \beta, \\ y + 2\alpha - 2\beta, & \text{if } \beta \le y \le n \end{cases}$$

on $\{0, \ldots, n\}$, where $0 < \alpha < \beta < n$ and $\beta < 2\alpha$. The function σ has a global minimum at $y = 0$ and a local minimum at $y = \beta$. Furthermore, we define the cost function

$$f_{\alpha,\beta,n}(x) = \sigma_{\alpha,\beta,n}(|x|) \tag{6.9}$$

on the set $S = \{0,1\}^n$ of binary strings of length n, where $|x|$ denotes the number of 1-bits in $x \in \{0,1\}^n$. For minimizing $f(x)$, we apply a GHC where the random variable R takes the value 0 with probability $1 - p$ and the value ∞ with probability p, such that better neighbors are always accepted, whereas worse neighbors are only accepted with probability p. (For our special example, this can also be formulated in terms of SA.) A neighbor solution to x is obtained by flipping a single bit in x. As subset i of the state space, we consider all $x \in S$ with $|x| = i$. Figure 6.1 shows the plot of the expected first hitting time (assuming random initial states) in dependence of the parameter p for the special case $\alpha = 8$, $\beta = 11$ and $n = 15$, computed by means of the formulas in [50]. The exploration-exploitation tradeoff is clearly seen: Both for large p, where exploration dominates, and for small p, where exploitation dominates, the runtime behavior of the GHC is suboptimal. For a certain $p = p^*$, the expected first hitting time is minimized; in our example, this happens for $p^* \approx 0.215$, resulting in $E(T_1) \approx 234.3$. Note that we have set the parameter p to a fixed, time-independent value; gradually decreasing it would possibly produce improved performance.[8]

Unfortunately, it is rather the exception than the regular case that after a decomposition of the space of memory states into subsets, the Markov property remains valid for the subsets. The *level reaching* method for obtaining runtime bounds, which has been developed in articles on evolutionary algorithms (EAs) (cf. Droste et al. [12] or Borisovsky and Eremeev [6]), is applicable in a broader range of cases. The basic idea of this method is to define *subgoals* for the final goal of attaining a set $\mathcal{M}^*$ of memory states producing optimal solutions (or solutions considered as of sufficient quality). In our generic framework, the method can be formulated as follows: Let $\mathcal{M}$ be the set of all possible memory states. A hierarchy

$$\mathcal{H}_1 \supseteq \mathcal{H}_2 \supseteq \ldots \supseteq \mathcal{H}_K = \mathcal{M}^*$$

is defined, where $\mathcal{H}_k \subseteq \mathcal{M}$ stands for the set of all memory states on a certain "quality level" k, and reaching this quality level is considered as "subgoal" k ($k = 1, \ldots, K$). The algorithm under consideration must fulfill the *monotonicity* constraint $m_t \in \mathcal{H}_k \Rightarrow m_{t+1} \in \mathcal{H}_k$. The expected first hitting time

[8] For the runtime analysis of SA with decreasing temperature, cf. Wegener [62].

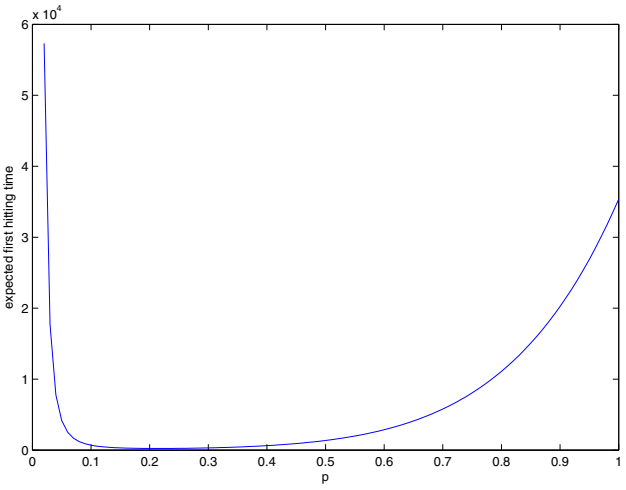

Fig. 6.1 Expected first hitting time of the GHC in dependence of the acceptance probability p for the illustration example.

t_k of subset $\mathcal{H}_k$, i.e., the expected time until subgoal k is reached for the first time, is given by $t_k = E(\min\{t : m_t \in \mathcal{H}_k\})$.

In the analysis of a considered algorithm on a special problem instance, one tries to identify upper bounds η_{ij} for the expected runtime until satisfying subgoal j, provided that the algorithm starts in an arbitrary memory state $m \in \mathcal{H}_i$. Then, e.g., it holds that

$$t_K \leq \sum_{i=1}^{K-1} \eta_{i,i+1}. \tag{6.10}$$

Often, the algorithm performs *independent* trials to achieve $\mathcal{H}_{i+1}$ from $\mathcal{H}_i$. Let in this situation $\alpha_{ij} > 0$ be a lower bound for the probability that $\mathcal{H}_j$ is reached after one iteration if the current state is in $\mathcal{H}_i$. Then, we can set $\eta_{i,i+1} = 1/\alpha_{i,i+1}$ and apply (6.10). Borisovsky and Eremeev [6] also provide stronger bounds on the values $(t_1, \ldots, t_K)$ derived from the matrix $A = (\alpha_{ij})$ in this more specific context.

A natural definition of subgoals derives from the possible cost function values: Let $\phi_1 > \ldots > \phi_K$ be the cost function values in decreasing order. Assume that the current memory state m_t contains x_t^{bsf} as a component. By defining $\mathcal{H}_k$ as the set of all memory states m for which $f(x^{bsf}) \leq \phi_k$, the required monotonicity property is satisfied. This principle has enabled the derivation of several runtime results for EAs (cf. [61, 51]).

Recently, it has been shown that the level-reaching approach can also be extended to *continuous* memory state spaces, as they occur in ACO, EDAs

or PSO. The article [31] provides general lemmas to ensure the mathematical validity of this extension, and derives expected first hitting time results for the MMAS variant of ACO on several standard test functions analyzed previously in the EA field.

(3) Martingales and Supermartingales

Let us consider again the Markov process (m_t) and assume that the currently proposed solution x_t is derived from the current state $m_t \in \mathcal{M}$, i.e., $x_t = \psi(m_t)$ with some function ψ. Furthermore, let us assume that a distance function d on $\mathcal{M}$ is given, which allows it in particular to define the distance $d(m, \mathcal{M}^*) = \min\{d(m, m^*) | m^* \in \mathcal{M}^*\}$ of a state m to the set $\mathcal{M}^*$ of memory states producing an optimal solution as the currently proposed solution. A possible way to define d may, e.g., consist in setting $d(m, m') = |f(\psi(m)) - f(\psi(m'))|$. For given $\mathcal{M}^*$, let us abbreviate $d(m, \mathcal{M}^*)$ by $d(m)$. We restrict the discussion to the case where $\mathcal{M}$ is a discrete finite set. He and Yao [36, 38] define the one-step *mean drift* in state m as the conditional expectation

$$E(d(m_t) - d(m_{t+1}) \,|\, m_t = m) = d(m) - \sum_{m'} p(m, m') d(m'),$$

where $p(m, m')$ is the transition probability from state m to state m'. In the case where the mean drift is always zero, the process $(d(m_t))$ is a *martingale*, which is a stochastic process (Y_t) with the property $E(Y_{t+1}|Y_1, \ldots, Y_t) = Y_t$. For problem instances, however, that are not *deceptive* in the sense of systematically misleading the search, it can happen that the drift is always positive. In this case, $(d(m_t))$ becomes a *supermartingale*, i.e., a process (Y_t) with $E(Y_{t+1}|Y_1, \ldots, Y_t) \leq Y_t$.

In [36, 38], drift analysis is applied to the runtime analysis of EAs on some special functions, and general conditions for classifying a problem instance as "easy" or "hard" for the considered algorithm are given, where "easy" and "hard" mean expected first hitting times of polynomial and exponential order in the problem instance size n, respectively. E.g., in [38], the following theorem is shown: A problem belongs to the "easy" class if and only if there exists a distance function $d(m)$ such that (i) $d(m_t) \leq g_1(n)$ with a polynomial $g_1(n)$, and $E(d(m_t) - d(m_{t+1}) \,|\, m_t) \geq c_{low}$ with a constant $c_{low} > 0$ for any state m_t in any iteration t.

(4) ODE Approximations

The scenario of a continuous memory state space $\mathcal{M}$ causes particular problems for the analysis, especially in the case where the search mechanism does not rely on the best-so-far solution x_t^{bsf}, but on other parts of the current state, such that in a natural definition of the sets $\mathcal{H}_k$ defining subgoals, the monotonicity property mentioned above is not satisfied anymore. An example is the Ant System variant of ACO, introduced in [10]. In the analysis of this algorithm, it is not clear how to define subgoals in a helpful way.

A tool to enable a mathematical runtime analysis also in such cases is the asymptotic approximation of the stochastic process by a limiting process obtained by letting some parameter tend to a boundary value. In the Ant System case, this can be done for the evaporation rate ρ, as shown in [25, 27]: If $\rho \to 0$ (which is a meaningful asymptotic for ACO in view of one of the convergence results outlined in Section 6.4 which has $\rho_t \to 0$), the Ant System process approaches a limiting process where the pheromone vector (i.e., the current memory state) follows a *deterministic* dynamic, described by a system of *ordinary differential equations* (ODEs). On the other hand, the sample points in L_t still remain stochastic, which means that the explorative capacity of the algorithm is not reduced. Conclusions on convergence and on expected first hitting time for special test functions can be derived, see [25, 27]. A similar approach has been presented in [52]. This technique is still relatively new in the literature on analysis of metaheuristics, such that its potential will yet have to be explored in the future.

6.7 Conclusions

In this paper, several notions of convergence in the context of optimization by metaheuristics have been discussed, and some fundamental mathematical proof ideas for showing convergence of special metaheuristic algorithms for all CO problems have been presented. In particular, we have distinguished between a weaker form of convergence termed best-so-far convergence, and a stronger form termed model convergence. Examples of algorithms owing one or both of these types of convergence properties have been given.

Let us shortly outline some open research topics. Several metaheuristic variants have not even been shown to converge in a best-so-far sense; as an example, let us mention ACO variants without pheromone bounds and working only with iteration-best reinforcement. It would be interesting to find out under which conditions these variants converge to the optimum at all. An obvious other topic of future research is of course to strengthen existing convergence results of best-so-far type, as available for a large class of other metaheuristic variants, to results of model convergence type.

Another open problem has already been outlined at the end of Subsection 6.4.2: the investigation of the connections between convergent or "subconvergent" parameter schemes for an algorithm to its performance within finite time (measured, e.g., by the average achieved solution quality).

In the field of convergence speed analysis, it seems that at the moment, the area of open problems is much larger that that of available results. Besides the challenging goal of analyzing the performance of metaheuristics on NP-complete problems, it may also be important to strive for more *general* results than those available at present. Although no-free-lunch theorems (cf. Section 6.6) set limits to generalizability, one might attempt to identify

runtime-relevant properties common to larger classes of different problems (e.g., fitness landscape properties) and to study the impacts of these properties on the runtime behavior in depth.

A final remark concerns "mat-heuristic" algorithms combining a metaheuristic approach with mathematical programming techniques. At the moment, convergence results for such algorithms (as an example, let us mention the well-known Local Branching algorithm by Fischetti and Lodi [16]) seem to be yet unavailable. As mentioned in the introduction, a rigorous mathematical analysis of algorithms would be especially desirable in this field overlapping with that of traditional mathematical optimization.

References

1. E.H.M. Aarts and J.H.L. Korst. *Simulated Annealing and Boltzmann Machines*. John Wiley and Sons, Chichester, UK, 1990.
2. M.H. Alrefaei and S. Andradóttir. A simulated annealing algorithm with constant temperature for discrete stochastic optimization. *Management Science*, 45:748–764, 1999.
3. L. Bianchi, M. Dorigo, L.M. Gambardella, and W.J. Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, to appear.
4. M. Birattari, P. Balaprakash, and M. Dorigo. The ACO/F-Race algorithm for combinatorial optimization under uncertainty. In K.F. Doerner, M. Gendreau, P. Greistorfer, W.J. Gutjahr, R.F. Hartl, and M. Reimann, editors, *Metaheuristics—Progress in Complex Systems Optimization*, pages 189–203. Springer Verlag, Berlin, Germany, 2006.
5. Y. Borenstein and R. Poli. Information perspective of optimization. In T.P. Runarsson, H.-G. Beyer, E.K. Burke, J.J. Merelo Guervós, L.D. Whitley, and X. Yao, editors, *Proceedings of the 9th Conference on Parallel Problem Solving from Nature*, volume 4193 of *Lecture Note in Computer Science*, pages 102–111. Springer Verlag, Berlin, Germany, 2006.
6. P.A. Borisovsky and A.V. Eremeev. A study on the performance of the (1+1)-evolutionary algorithm. In K.A. De Jong, R. Poli, and J.E. Rowe, editors, *Proceedings of Foundations of Genetic Algorithms*, volume 7, pages 271–287. Morgan Kaufmann Publishers, San Mateo, CA, 2003.
7. J. Brimberg, P. Hansen, and N. Mladenović. Convergence of variable neighborhood search. Les Cahiers du GERAD G-2002-21, Groupe d'études et de recherche en analyse des décisions (GERAD), Montréal, Canada, 2002.
8. T. Homem de Mello. Variable-sample methods for stochastic optimization. *ACM Transactions on Modeling and Computer Simulation*, 13:108–133, 2003.
9. F. Van den Bergh and A.P. Engelbrecht. A study of particle swarm optimization particle trajectories. *Information Sciences*, 176:937–971, 2006.
10. M. Dorigo, V. Maniezzo, and A. Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics*, 26:1–13, 1996.
11. S. Droste, T. Jansen, and I. Wegener. Perhaps not a free lunch but at least a free appetizer. In W. Banzhaf, J.M. Daida, A.E. Eiben, M.H. Garzon, V. Honavar, M.J. Jakiela, and R.E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference 1999*, pages 833–839. Morgan Kaufmann, San Mateo, CA, 1999.
12. S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276:51–81, 2002.

13. S. Droste, T. Jansen, and I. Wegener. Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems*, 39(4):525–544, 2006.
14. T. English. Optimization is easy and learning is hard in the typical function. In *Proceedings of the 2000 Congress on Evolutionary Computation*, volume 2, pages 924–931. IEEE Press, Piscataway, NJ, 2000.
15. T. English. On the structure of sequential search: beyond no free lunch. In J. Gottlieb and G.R. Raidl, editors, *Evolutionary Computation in Combinatorial Optimization, 4th European Conference, EvoCOP 2004*, volume 3004 of *Lecture Notes in Computer Science*, pages 95–103. Springer Verlag, Berlin, Germany, 2004.
16. M. Fischetti and A. Lodi. Local branching. *Mathematical Programming Ser. B*, 98:23–47, 2003.
17. S.B. Gelfand and S.K. Mitter. Analysis of simulated annealing for optimization. In *Proceedings of the 24th IEEE Conference on Decision and Control*, pages 779–786, 1985.
18. S.B. Gelfand and S.K. Mitter. Simulated annealing with noisy or imprecise measurements. *Journal of Optimization Theory and Applications*, 69:49–62, 1989.
19. F. Glover and G. Kochenberger, editors. *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*. Springer, 2003.
20. C. Gonzalez, J.A. Lozano, and P. Larrañaga. Analyzing the PBIL algorithm by means of discrete dynamical systems. *Complex Systems*, 11:1–15, 1997.
21. W.J. Gutjahr. A graph–based ant system and its convergence. *Future Generation Computer Systems*, 16:873–888, 2000.
22. W.J. Gutjahr. ACO algorithms with guaranteed convergence to the optimal solution. *Information Processing Letters*, 82:145–153, 2002.
23. W.J. Gutjahr. A converging ACO algorithm for stochastic combinatorial optimization. In A.A. Albrecht and K. Steinhöfel, editors, *Stochastic Algorithms: Foundations and Applications, Second International Symposium, SAGA 2003*, volume 2827 of *Lecture Notes in Computer Science*, pages 10–25. Springer Verlag, Berlin, Germany, 2003.
24. W.J. Gutjahr. An ant-based approach to combinatorial optimization under uncertainty. In M. Dorigo, L. Gambardella, F. Mondada, T. Stützle, M. Birratari, and C. Blum, editors, *ANTS'2004, Fourth International Workshop on Ant Algorithms and Swarm Intelligence*, volume 3172 of *Lecture Notes in Computer Science*, pages 238–249. Springer Verlag, Berlin, Germany, 2004.
25. W.J. Gutjahr. On the finite-time dynamics of ant colony optimization. *Methodology and Computing in Applied Probability*, 8:105–133, 2006.
26. W.J. Gutjahr. Mathematical runtime analysis of ACO algorithms: survey on an emerging issue. *Swarm Intelligence*, 1:59–79, 2007.
27. W.J. Gutjahr. First steps to the runtime complexity analysis of ant colony optimization. *Computers & Operations Research*, 35:2711–2727, 2008.
28. W.J. Gutjahr. Stochastic search in metaheuristics. Technical report, Department of Statistics and Decision Support Systems, University of Vienna, 2008.
29. W.J. Gutjahr, S. Katzensteiner, and P. Reiter. A VNS algorithm for noisy problems and its application to project portfolio analysis. In J. Hromkovic, R. Královic, M. Nunkesser, and P. Widmayer, editors, *Stochastic Algorithms: Foundations and Applications, Second International Symposium, SAGA 2007*, volume 4665 of *Lecture Notes in Computer Science*, pages 93–104. Springer Verlag, Berlin, Germany, 2007.
30. W.J. Gutjahr and G. Pflug. Simulated annealing for noisy cost functions. *Journal of Global Optimization*, 8:1–13, 1996.
31. W.J. Gutjahr and G. Sebastiani. Runtime analysis of ant colony optimization with best-so-far reinforcement. *Methodology and Computing in Applied Probability*, 10:409–433, 2008.
32. B. Hajek. Cooling schedules for optimal annealing. *Mathematics of Operations Research*, 13:311–329, 1988.

33. P. Hansen and N. Mladenović. Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, 130:449–467, 2001.
34. R.F. Hartl. A global convergence proof for a class of genetic algorithms. Technical report, Institut für Ökonometrie & Operations Research, Technische Universität Wien, 1990.
35. J. He and X. Yao. Conditions for the convergence of evolutionary algorithms. *Journal of Systems Architecture*, 47:601–612, 2001.
36. J. He and X. Yao. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127:57–85, 2003.
37. J. He and X. Yao. Towards an analytic framework for analysing the computation time of evolutionary algorithms. *Artificial Intelligence*, 145:59–97, 2003.
38. J. He and X. Yao. A study of drift analysis for estimating computation time of evolutionary algorithms. *Natural Computing*, 3:21–35, 2004.
39. H.H. Hoos. On the runtime behavior of stochastic local search algorithms for SAT. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 661–666. AAAI Press / The MIT Press, Menlo Park, CA, USA, 1999.
40. H.H. Hoos and T. Stützle. Local search algorithms for SAT: an empirical investigation. *Journal of Automated Reasoning*, 24:421–481, 2000.
41. C. Igel and M. Toussaint. On classes of functions for which no free lunch results hold. *Information Processing Letters*, 86:317–321, 2003.
42. C. Igel and M. Toussaint. A no-free-lunch theorem for non-uniform distributions of target functions. *Journal of Mathematical Modelling and Algorithms*, 3:313–322, 2004.
43. S.H. Jacobson, K.A. Sullivan, and A.W. Johnson. Discrete manufacturing process design optimization using computer simulation and generalized hill climbing algorithms. *Engineering Optimization*, 31:147–260, 1998.
44. S.H. Jacobson and E. Yuecesan. Analyzing the performance of generalized hill climbers. *Journal of Heuristics*, 10:387–405, 2004.
45. J. Jaegerskuepper. Lower bonds for hit-and-run direct search. In J. Hromkovic, R. Královic, M. Nunkesser, and P. Widmayer, editors, *Stochastic Algorithms: Foundations and Applications, Second International Symposium, SAGA 2007*, volume 4665 of *Lecture Notes in Computer Science*, pages 118–129. Springer Verlag, Berlin, Germany, 2007.
46. Y. Jin and J. Branke. Evolutionary optimization in uncertain environments—a survey. *IEEE Transactions on Evolutionary Computation*, 9:303–317, 2005.
47. J. Kennedy and R.C. Eberhart. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks*, pages 1942–1948. IEEE Press, Piscataway, NJ, 1995.
48. J. Kennedy and R.C. Eberhart. A discrete binary version of the particle swarm algorithm. In *Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics*, volume 5, pages 4104–4109. IEEE Press, Piscataway, NJ, 1997.
49. L. Margolin. On the convergence of the cross-entropy method. *Annals of Operations Research*, 134:201–214, 2005.
50. H. Muehlenbein and J. Zimmermann. Size of neighborhood more important than temperature for stochastic local search. In *Proceedings of the 2000 Congress on Evolutionary Computation*, volume 2, pages 1017–1024. IEEE Press, Piscataway, NJ, 2000.
51. P.S. Oliveto, J. He, and X. Yao. Time complexity of evolutionary algorithms for combinatorial optimization: a decade of results. *International Journal of Automation and Computing*, 4:281–293, 2007.
52. P. Purkayastha and 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*, pages 340–345. IEEE Press, Piscataway, NJ, 2007.
53. R.Y. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, pages 127–170, 1999.
54. G. Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE Transactions on Neural Networks*, 5:96–101, 1994.

55. G. Sebastiani and G.L. Torrisi. An extended ant colony algorithm and its convergence analysis. *Methodology and Computing in Applied Probability*, 7:249–263, 2005.
56. J.C. Spall, S.D. Hill, and D.R. Stark. Theoretical framework for comparing several stochastic optimization algorithms. In G. Calafiore and F. Dabbene, editors, *Probabilistic and Randomized Methods for Design under Uncertainty*, pages 99–117. Springer Verlag, London, UK, 2006.
57. T. Stützle and M. Dorigo. A short convergence proof for a class of ACO algorithms. *IEEE Transactions on Evolutionary Computation*, 6:358–365, 2002.
58. T. Stützle and H.H. Hoos. $\mathcal{MAX} - \mathcal{MIN}$ ant system. *Future Generation Computer Systems*, 16:889–914, 2000.
59. A.S. Thikomirov. On the convergence rate of the Markov homogeneous monotone optimization method. *Computational Mathematics and Mathematical Physics*, 47:817–828, 2007.
60. I.C. Trelea. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information Processing Letters*, 85:317–325, 2003.
61. I. Wegener. Methods for the analysis of evolutionary algorithms on pseudo-boolean functions. In R. Sarker, M. Mohammadia, and X. Yao, editors, *Evolutionary Optimization*, volume 48 of *International Series in Operations Research & Management Science*. Kluwer Academic Publishers, Norwell, MA, 2003.
62. I. Wegener. Simulated annealing beats metropolis in combinatorial optimization. In L. Caires, G.F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005*, volume 3580 of *Lecture Notes in Computer Science*, pages 589–601. Springer Verlag, Berlin, Germany, 2005.
63. D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1:67–82, 1997.
64. Y. Yu and Z.-H. Zhou. A new approach to estimating the expected first hitting time of evolutionary algorithms. In *Proceedings of the Twentyfirst National Conference on Artificial Intelligence*, pages 555–560. AAAI Press / The MIT Press, Menlo Park, CA, USA, 2006.
65. M. Zlochin, M. Birattari, N. Meuleau, and M. Dorigo. Model-based search for combinatorial optimization: a critical survey. *Annals of Operations Research*, 131:373–379, 2004.