Annals of Operations Research 131, 373–395, 2004

Model-Based Search for Combinatorial Optimization: A Critical Survey

MARK ZLOCHIN* zmark@weizmann.ac.il
Dept. of Applied Mathematics and Computer Science, The Weizmann Institute of Science, Israel

MAURO BIRATTARI* mbiro@ulb.ac.be
IRIDIA, Université Libre de Bruxelles, Brussels, Belgium

NICOLAS MEULEAU* nmeuleau@email.arc.nasa.gov
NASA Ames Research Center, Mail stop 269-2, Moffett Field, CA, USA

MARCO DORIGO mdorigo@ulb.ac.be
IRIDIA, Université Libre de Bruxelles, Brussels, Belgium

Abstract. In this paper we introduce *model-based search* as a unifying framework accommodating some recently proposed metaheuristics for combinatorial optimization such as ant colony optimization, stochastic gradient ascent, cross-entropy and estimation of distribution methods. We discuss similarities as well as distinctive features of each method and we propose some extensions.

Keywords: ant colony optimization, cross-entropy method, stochastic gradient ascent, estimation of distribution algorithms, adaptive optimization, metaheuristics

1. Introduction

The necessity to solve $\mathcal{NP}$-hard optimization problems, for which the existence of efficient exact algorithms is highly unlikely, has led to a wide range of heuristic algorithms that implement some sort of search in the solution space. These heuristic algorithms can be classified, similarly to what is done in the machine learning field (Quinlan, 1993), as being either *instance-based* or *model-based*. Most of the classical search methods may be considered instance-based, since they generate new candidate solutions using solely the current solution or the current "population" of solutions. Typical representatives of this class are genetic algorithms (Holland, 1975) or local search and its variants, such as, for example, simulated annealing and iterated local search (Aarts and Lenstra, 1997). On the other hand, in the last decade several new methods, which may be classified as *model-based search* (MBS) algorithms, have been proposed. In model-based search algorithms, candidate solutions are generated using a parameterized probabilistic model that is up-

* This work was carried out while Mark Zlochin and Nicolas Meuleau were with IRIDIA, Université Libre de Bruxelles, Brussels, Belgium, and Mauro Birattari was with Intellektik, Technische Universität Darmstadt, Darmstadt, Germany.

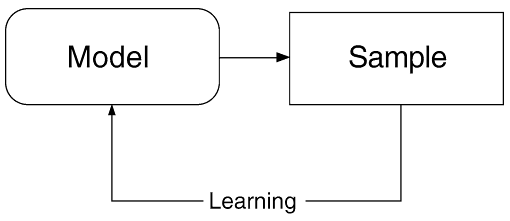

Figure 1. Schematic description of the MBS approach.

dated using the previously seen solutions in such a way that the search will concentrate in the regions containing high quality solutions. In order to avoid any terminological confusion, we would like to emphasize that the term "model" is used here to denote an adaptive stochastic mechanism for generating candidate solutions, and not an approximate description of the environment,[1] as done, for example, in reinforcement learning (Sutton and Barto, 1998). The general approach is described schematically in figure 1. Some of the early works exploiting the model-based approach, such as ant colony optimization (Dorigo, 1992; Dorigo, Maniezzo, and Colorni, 1996; Dorigo and Di Caro, 1999) and population-based incremental learning (Baluja and Caruana, 1995), do not provide an explicit description of the model-based idea. The first explicit description of a solution process consisting in a series of suitably updated probability distributions on the solution space was given by De Bonet, Isbell, and Viola (1997). More recently, on the basis of concepts borrowed from the stochastic simulation field and, in particular, from rare events estimation, Rubinstein (1999a) re-proposed the ideas of De Bonet and co-workers and provided an extensive analysis of many details (De Boer et al., 2001).

While the behavior of classical instance-based search methods has been thoroughly investigated and is relatively well understood, the model-based search field is still little more than a collection of independently developed heuristic techniques, without solid theoretical foundations. The goal of this paper is to provide a unifying framework that accommodates all these seemingly unrelated methods and to analyze their similarities as well as their distinctive features. The analysis of these methods within a common framework allows to discriminate between the essential elements of the algorithm and those that appear only for historical reasons.

A well-established approach that belongs to the MBS framework is the *ant colony optimization* (ACO) metaheuristic (Dorigo, 1992; Dorigo, Maniezzo, and Colorni, 1996; Dorigo and Di Caro, 1999). The distinctive feature of ant colony optimization is a particular type of probabilistic model, in which a structure called *construction graph* is coupled with a set of stochastic procedures called *artificial ants*. The artificial ants have a two-fold function – they both generate solutions and update the model's parameters. Various model update rules have been proposed within the ACO framework, but they are all of a somewhat heuristic nature and are lacking a theoretical justification.

On the other hand, the *stochastic gradient ascent* (SGA) (Robbins and Monro, 1951; Bertsekas, 1995) and the *cross-entropy* (CE) (Rubinstein, 1999a) methods provide a systematic way for the derivation of model update rules in the MBS framework, without being restricted to a particular type of probabilistic model. As we show in the

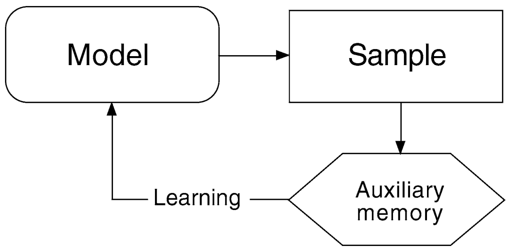

Figure 2. The MBS with auxiliary memory.

following, both the stochastic gradient ascent and the cross-entropy methods can be cast into the ACO framework, and, in fact, in some cases the cross-entropy method leads to the same update rule as does stochastic gradient ascent. Moreover, quite unexpectedly, some existing ACO updates may be re-derived as a particular implementation of the cross-entropy method.

It should be noticed that figure 1 describes the MBS approach in its "pure" form, where the model update is based solely on the current solutions' sample. However, many model-based search algorithms update the model using not only the current sample, but also some additional information gathered during the search and stored in an auxiliary memory, as described in figure 2. In particular, a recently developed class of evolutionary algorithms called *estimation of distribution algorithms* (EDAs) (Pelikan, Goldberg, and Lobo, 1999; Larrañaga and Lozano, 2001) may be considered a particular realization of model-based search with an auxiliary memory that stores high-quality solutions encountered during the search. Not only all these algorithms belong to the MBS approach, but many of them are actually closely related to the ACO and CE frameworks, as we show in the following.

The paper is structured as follows. In section 2 we describe model-based search in general terms and present stochastic gradient ascent and cross-entropy as particular realizations of the MBS approach. The relationship between the two methods is also discussed in that section.

Section 3 presents the ACO metaheuristic and discusses the implementation of the cross-entropy and the stochastic gradient ascent methods using the ACO-type construction mechanism as a model.

In section 4 the estimation of distribution algorithms are presented as a particular realization of MBS with auxiliary memory. An overview of existing EDAs is given and their relations to the ACO framework and the cross-entropy method are discussed.

Section 5 draws some conclusions and outlines several interesting future research directions.

2. Model-based search

Let us consider a minimization problem[2] $(\mathcal{S}, f)$, where $\mathcal{S}$ is the *set of feasible solutions*, f is the *objective function*, which assigns to each solution $s \in \mathcal{S}$ a cost $f(s)$. The goal of the minimization problem is to find an optimal solution s^*, that is, a feasible solution of minimum cost. The set of all optimal solutions is denoted by $\mathcal{S}^*$.

At a very general level, the model-based search approach attempts to solve this minimization problem by repeating the following two steps:

– Candidate solutions are constructed using some parameterized probabilistic model, that is, a parameterized probability distribution over the solution space.
– The candidate solutions are used to modify the model[3] in a way that is deemed to bias future sampling toward low cost solutions.

As it was already mentioned in the introduction, one may also use an auxiliary memory, in which some important information collected during the search is stored. The memory, which may store, for example, information on the distribution of the cost values or a collection of high-quality solutions, can be later used for the model update. Moreover, in some cases we may wish to build a new model at every iteration, rather than to iteratively update the same one.

For any algorithm belonging to this general scheme, two components, corresponding to the two steps above, need to be instantiated:

– A probabilistic model that allows an efficient generation of the candidate solutions.
– An update rule for the model's parameters and/or structure.

In the remainder of this section we discuss two systematic approaches within the MBS framework, namely stochastic gradient ascent and cross-entropy methods, which define the second component, that is the update rule for the model. We show that, although having a completely different motivation, the two approaches are closely related. In fact, we show that a particular version of CE produces the same updates as SGA does.

Throughout the remainder of this section we assume that a space $\mathcal{M}$ of possible probabilistic models is given and that it is expressive enough. Specifically, we need to assume that for every possible solution s, the distribution $\delta_s(\cdot)$ (defined as $\delta_s(s') = 1$, if $s' = s$, and $\delta_s(s') = 0$ otherwise) belongs to $\mathcal{M}$. This condition may actually be relaxed by assuming instead that δ_s is in the closure of $\mathcal{M}$, that is, that there exists a sequence $P_i \in \mathcal{M}$ for which $\lim_{i\to\infty} P_i = \delta_s$. This "expressiveness" assumption is needed in order to ensure that the sampling can concentrate in the proximity of any solution, the optimal solution in particular.

2.1. *Stochastic gradient ascent*

Let us assume that the model structure is fixed, and the model space, $\mathcal{M}$, is smoothly parameterized by $\mathcal{T} \in \Phi \subset \mathbb{R}^m$, where Φ is an m-dimensional parameter space. In other words, $\mathcal{M} = \{P_{\mathcal{T}}(\cdot) \mid \mathcal{T} \in \Phi\}$ and for any $s \in \mathcal{S}$ the function $P_{\mathcal{T}}(s)$ is smooth[4] with respect to $\mathcal{T}$.

The original optimization problem may be replaced with the following equivalent continuous *maximization problem*:

$$\mathcal{T}^* = \operatorname*{argmax}_{\mathcal{T}} \mathcal{E}(\mathcal{T}), \tag{1}$$

where $\mathcal{E}(\mathcal{T}) = E_{\mathcal{T}} Q_f(s)$, $E_{\mathcal{T}}$ denotes expectation with respect to $P_{\mathcal{T}}$, and $Q_f(s)$ is a fixed *quality function*, which is strictly decreasing with respect to f, that is, $Q_f(s_1) < Q_f(s_2) \Leftrightarrow f(s_1) > f(s_2)$.

It may be easily verified that, under the "expressiveness" assumption we made about the model space, the support of $P_{\mathcal{T}^*}$ (i.e., the set $\{s \mid P_{\mathcal{T}^*}(s) > 0\}$) is necessarily contained in $\mathcal{S}^*$. This implies that solving problem (1) is equivalent to solving the original combinatorial optimization problem.

One may then search for an optimum (possibly a local one) of the problem given by equation (1) using a gradient ascent method (in other words, gradient ascent may be used as a heuristic to change $\mathcal{T}$ with the goal of solving equation (1)):

– Start from some initial guess $\mathcal{T}^0$.
– At stage t, calculate the gradient $\nabla\mathcal{E}(\mathcal{T}^t)$ and update $\mathcal{T}^{t+1}$ to be $\mathcal{T}^t + \alpha_t \nabla\mathcal{E}(\mathcal{T}^t)$, where α_t is a step-size parameter.

The gradient can be calculated (bearing in mind that $\nabla \ln f = \nabla f / f$) as follows:

$$\begin{aligned}\nabla\mathcal{E} &= \nabla E_{\mathcal{T}} Q_f(s) = \nabla \sum_s Q_f(s) P_{\mathcal{T}}(s) = \sum_s Q_f(s) \nabla P_{\mathcal{T}}(s)\\ &= \sum_s P_{\mathcal{T}}(s) Q_f(s) \frac{\nabla P_{\mathcal{T}}(s)}{P_{\mathcal{T}}(s)} = \sum_s P_{\mathcal{T}}(s) Q_f(s) \nabla \ln P_{\mathcal{T}}(s)\\ &= E_{\mathcal{T}} Q_f(s) \nabla \ln P_{\mathcal{T}}(s).\end{aligned} \tag{2}$$

However, the gradient ascent algorithm cannot be implemented in practice, as for its evaluation a summation over the whole search space is needed. A more practical alternative would be to use *stochastic gradient ascent* (Robbins and Monro, 1951; Bertsekas, 1995), which replaces the expectation in equation (2) by an empirical mean of a sample generated from $P_{\mathcal{T}}$.

The update rule for the stochastic gradient is:

$$\mathcal{T}^{t+1} = \mathcal{T}^t + \alpha_t \sum_{s \in S_t} Q_f(s) \nabla \ln P_{\mathcal{T}^t}(s), \tag{3}$$

where S_t is the sample at iteration t.

In order to derive a practical algorithm from the SGA approach, we need a model for which the derivatives of the $\ln P_{\mathcal{T}}(\cdot)$ can be calculated efficiently. In section 3.3 we show how this can be done in the context of the iterative construction scheme used in the ACO metaheuristic.

2.2. Cross-entropy method

The basic ideas behind the cross-entropy (CE) method for combinatorial optimization can be already found in De Bonet, Isbell, and Viola (1997). However, the full development of the method was given in the works of Rubinstein and co-workers, who have initially proposed this method as a tool for rare events estimation in stochastic simulation

(Rubinstein, 1999b; Lieber, 1999) and have later adapted it to the field of combinatorial optimization (Rubinstein, 1999a, 2001). In this overview we focus on the central idea of cross-entropy and we propose a presentation of the main concepts without reference to rare events estimation. This presentation should appear more straightforward to the operations research community.[5]

Starting from some initial distribution $P_0 \in \mathcal{M}$, the CE method inductively builds a series of distributions $P_t \in \mathcal{M}$, in an attempt to increase the probability of generating low-cost solutions after each iteration. A tentative way to achieve this goal is to set P_{t+1} equal to

$$\widehat{P} \propto P_t Q_f, \tag{4}$$

where Q_f is, again, some quality function, depending on the cost value.

If this were possible, then the multiplication by Q_f would bias the probability toward high-quality solutions and, for time independent quality functions,[6] after n iteration we would obtain $P_n \propto P_0(Q_f)^n$. Consequently, as $n \to \infty$, P_n would converge to a probability distribution restricted to $\mathcal{S}^*$. Unfortunately, even if the distribution P_t belongs to the family $\mathcal{M}$, the distribution $\widehat{P}$ as defined by equation (4) does not necessarily remain in $\mathcal{M}$,[7] hence some sort of projection is needed.

Accordingly, a natural candidate for P_{t+1} is the distribution $P \in \mathcal{M}$ that minimizes the *Kullback–Leibler divergence* (Kullback, 1959), which is a commonly used measure of misfit between two distributions:

$$D(\widehat{P} \| P) = \sum_s \widehat{P}(s) \ln \frac{\widehat{P}(s)}{P(s)}, \tag{5}$$

or equivalently the *cross-entropy*:

$$-\sum_s \widehat{P}(s) \ln P(s). \tag{6}$$

Since $\widehat{P} \propto P_t Q_f$, the cross-entropy minimization is equivalent to the following maximization problem:

$$P_{t+1} = \operatorname*{argmax}_{P \in \mathcal{M}} \sum_s P_t(s) Q_f(s) \ln P(s). \tag{7}$$

It should be noted that in the cross-entropy method, differently from what done by SGA, the quality function is only required to be non-increasing with respect to the cost and may also be time-dependent, either deterministically or stochastically. For example, it might depend on the points sampled so far. One common choice is $Q^t_f(s) = I(f(s) < f_t)$, where $I(\cdot)$ is an indicator function, and f_t is, for example, some quantile (e.g., lower 10%) of the cost distribution during the last iteration[8]. Another quality function considered in Rubinstein (1999a) is a Boltzmann function $Q_f(s) = \exp(-f(s)/\gamma)$, where γ is changed adaptively based on the sample.

Similarly to the gradient ascent algorithm, the maximization problem given by equation (7) cannot be solved in practice, as the evaluation of the function

$\sum_s P_t(s) Q_f(s) \ln P(s)$ requires summation over the whole solution space, and once again a finite sample approximation is used instead:

$$P_{t+1} = \operatorname*{argmax}_{P \in \mathcal{M}} \sum_{s \in S_t} Q_f(s) \ln P(s), \tag{8}$$

where S_t is a sample from P_t.

Note that if the quality function is of the form $I(f(s) < c)$, then equation (8) defines a *maximum-likelihood* model, with the sample used for estimation being restricted to the top-quality solutions. With other quality functions, equation (8) may be interpreted as defining a weighted maximum-likelihood estimate.

In some relatively simple cases, some of which are discussed in sections 3 and 4, the problem given by equation (8) can be solved exactly. In general, however, the analytical solution is unavailable. Still, even if the exact solution is not known, some iterative methods for solving this optimization problem may be used.

A natural candidate for the iterative solution of the maximization problem (8) is gradient ascent:

```
– Start with $\mathcal{T}' = \mathcal{T}^t$. (Other starting points are possible, but this is the most natural
  one, since we may expect $\mathcal{T}^{t+1}$ to be close to $\mathcal{T}^t$.)
– Repeat:
      $\mathcal{T}' \leftarrow \mathcal{T}' + \alpha \sum_{s \in S_t} Q_f(s) \nabla \ln P_{\mathcal{T}'}(s)$,
      where $\alpha$ is a step-size parameter
      Until some stopping criterion is satisfied.
– Set $\mathcal{T}^{t+1} = \mathcal{T}'$.
```

It should be noted that, since the new vector $\mathcal{T}^{t+1}$ is a random variable, depending on a sample, there is no use in running the gradient ascent process till full convergence. Instead, in order to obtain some robustness against sampling noise, we may use a fixed number of gradient ascent updates. One particular choice, which is of special interest, is the use of a single gradient ascent update, leading to the updating rule:

$$\mathcal{T}^{t+1} = \mathcal{T}^t + \alpha_t \sum_{s \in S_t} Q_f(s) \nabla \ln P_{\mathcal{T}^t}(s), \tag{9}$$

which is identical to the SGA update (equation (3)). However, as it was already mentioned earlier, the CE method imposes less restrictions on the quality function (e.g., allowing it to change over time), hence the resulting algorithm may be seen as a generalization of SGA.

As with SGA, in order to have an efficient algorithm, a model is needed for which the calculation of the derivatives can be carried out in reasonable time. In the next section, we show that this is indeed possible for the models typically used in ant colony optimization.

3. The ACO metaheuristic and the SGA/CE methods

So far, we have limited our discussion to the generic approaches for updating the model. However, this is only one out of the two components needed in any MBS algorithm. In order to complete the description of a MBS algorithm, a probabilistic model needs to be specified.

In this section we describe the ant colony optimization metaheuristic (Dorigo, 1992; Dorigo, Maniezzo, and Colorni, 1996; Dorigo and Di Caro, 1999) that employs a particular type of probabilistic model in which a structure called *construction graph* is coupled with a set of stochastic procedures called *artificial ants*. The artificial ants build solutions in an iterative manner using local information stored in the construction graph.[9] After describing the probabilistic model, we present several updates that were suggested in the past within the ant colony optimization framework as well as the ones derived from the stochastic gradient ascent algorithm and the cross-entropy method.

3.1. Ant colony optimization – the probabilistic model

We assume that the combinatorial optimization problem $(\mathcal{S}, f)$ is mapped on a problem that can be characterized by the following list of items:[10]

- A finite set $\mathcal{C} = \{c_1, c_2, \ldots, c_{N_C}\}$ of *components*, where N_C is the number of components.
- A finite set $\mathcal{X}$ of *states* of the problem, where a state is a sequence $x = \langle c_i, c_j, \ldots, c_k, \ldots \rangle$ over the elements of $\mathcal{C}$. The length of a sequence x, that is, the number of components in the sequence, is expressed by $|x|$. The maximum length of a sequence is bounded by a positive constant $n < +\infty$.
- The set of (candidate) solutions $\mathcal{S}$ is a subset of $\mathcal{X}$ (i.e., $\mathcal{S} \subseteq \mathcal{X}$).
- A set of feasible states $\widetilde{\mathcal{X}}$, with $\widetilde{\mathcal{X}} \subseteq \mathcal{X}$, defined via a set of *constraints* Ω.
- A non-empty set $\mathcal{S}^*$ of optimal solutions, with $\mathcal{S}^* \subseteq \widetilde{\mathcal{X}}$ and $\mathcal{S}^* \subseteq \mathcal{S}$.

Given the above formulation, artificial ants build candidate solutions by performing randomized walks on the completely connected, weighted graph $\mathcal{G} = (\mathcal{C}, \mathcal{L}, \mathcal{T})$, where the vertices are the components $\mathcal{C}$, the set $\mathcal{L}$ fully connects the components $\mathcal{C}$, and $\mathcal{T}$ is a vector gathering so-called *pheromone trails* τ.[11] The graph $\mathcal{G}$ is called *construction graph.*

Each artificial ant is put on a randomly chosen vertex of the graph and then it performs a randomized walk by moving at each step from vertex to vertex in the graph in such a way that the next vertex is chosen stochastically according to the strength of the pheromone currently on the arcs. While moving from one node to another of the graph $\mathcal{G}$, constraints Ω may be used to prevent ants from building infeasible solutions. Formally, the solution construction behavior of a generic ant can be described as follows:

ANT_SOLUTION_CONSTRUCTION

- for each ant:
  - select a start node c_1 according to some problem dependent criterion,
  - set $k = 1$ and $x_k = \langle c_1 \rangle$.
- While $x_k = \langle c_1, c_2, \ldots, c_k \rangle \in \widetilde{\mathcal{X}}$ and $x_k \notin \mathcal{S}$ and $J_{x_k} \neq \emptyset$ do:
  at each step k, after building the sequence x_k, select the next node (component) c_{k+1} randomly following

$$P_{\mathcal{T}}(c_{k+1} = c | x_k) = \begin{cases} \dfrac{F_{(c_k,c)}(\tau(c_k, c))}{\sum_{(c_k,y)\in J_{x_k}} F_{(c_k,y)}(\tau(c_k, y))} & \text{if } (c_k, c) \in J_{x_k}, \\ 0 & \text{otherwise;} \end{cases} \tag{10}$$

  where a connection (c_k, y) belongs to J_{x_k} iff the sequence $x_{k+1} = \langle c_1, c_2, \ldots, c_k, y \rangle$ satisfies the constraints Ω (that is, $x_{k+1} \in \widetilde{\mathcal{X}}$) and $F_{(i,j)}(z)$ is some monotonic function – most commonly, $z^\alpha \eta(i, j)^\beta$, where $\alpha, \beta > 0$ and η are heuristic "visibility" values (Dorigo, Maniezzo, and Colorni, 1996). If at some stage $x_k \notin \mathcal{S}$ and $J_{x_k} = \emptyset$, that is, the construction process has reached a dead-end, the current state x_k is discarded.[12]

For certain problems, one may find useful to use a more general scheme, where F depends on the pheromone values of several "related" connections, rather than just a single one. Moreover, instead of the *random-proportional rule* above, different selection schemes, such as the *pseudo-random-proportional rule* (Dorigo and Gambardella, 1997), may be considered.

The probabilistic rule given by equation (10), together with the underlying construction graph, implicitly defines a first component of the MBS algorithm – the probabilistic model. Having chosen the probabilistic model, the next step is to choose the parameter update mechanism. In the following, we describe several updates that were suggested in the past within the ant colony optimization framework as well as the ones derived from the stochastic gradient ascent algorithm and the cross-entropy method.

3.2. *Ant colony optimization – the pheromone updates*

Many different schemes for pheromone update have been proposed within the ACO framework. For an extensive overview, see Dorigo and Stützle (2002, 2004). Most pheromone updates can be described using the following generic scheme:

GENERIC_ACO_UPDATE

- $\forall s \in \widehat{S}_t, \forall (i, j) \in s \colon \tau(i, j) \leftarrow \tau(i, j) + Q_f(s | S_1, \ldots, S_t)$,
- $\forall (i, j) \colon \tau(i, j) \leftarrow (1 - \rho) \cdot \tau(i, j)$

where S_i is the sample in the ith iteration, ρ, $0 < \rho \leqslant 1$, is the evaporation rate, and $Q_f(s|S_1, \ldots, S_t)$ is some "quality function," which is typically required to be non-increasing with respect to f and is defined over the "reference set" $\widehat{S}_t$.

Different ACO algorithms may use different quality functions and reference sets. For example, in the very first ACO algorithm – Ant System (Dorigo, Maniezzo, and Colorni, 1991, 1996) – the quality function was simply $1/f(s)$ and the reference set $\widehat{S}_t = S_t$. In a more recently proposed scheme, called *iteration best update* (Dorigo and Gambardella, 1997), the reference set was a singleton containing the best solution within S_t (if there were several iteration-best solutions, one of them was chosen randomly). For the *global-best update* (Dorigo and Gambardella, 1997; Stützle and Hoos, 1997), the reference set contained the best among all the iteration-best solutions (and if there were more than one global-best solution, the earliest one was chosen). In Dorigo, Maniezzo, and Colorni (1996) an *elitist* strategy was introduced, in which the update was a combination of the previous two.

In case a good lower bound on the optimal solution cost is available, one may use the following quality function (Maniezzo, 1999):

$$Q_f(s|S_1, \ldots, S_t) = \tau_0\left(1 - \frac{f(s) - LB}{\bar{f} - LB}\right) = \tau_0\frac{\bar{f} - f(s)}{\bar{f} - LB}, \tag{11}$$

where $\bar{f}$ is the average of the costs of the last k solutions and LB is the lower bound on the optimal solution cost. With this quality function, the solutions are evaluated by comparing their cost to the average cost of the other recent solutions, rather than by using the absolute cost values. In addition, the quality function is automatically scaled based on the proximity of the average cost to the lower bound.

A pheromone update, which slightly differs from the generic update described above, was used in *ant colony system* (ACS) (Dorigo and Gambardella, 1997). There the pheromones are evaporated by the ants online during the solution construction, hence only the pheromones involved in the construction evaporate.

Two additional modifications of the generic update were described in the literature. The first one, introduced by Stützle and Hoos in their $\mathcal{MAX}$–$\mathcal{MIN}$ Ant System (1997, 2000), uses maximum and minimum pheromone trail limits. With this modification, the probability to generate any particular solution is kept above some positive threshold, which helps preventing search stagnation and premature convergence to suboptimal solutions.

The second modification, proposed under the name of *hyper-cube framework* for ACO (HCF-ACO) (Blum, Roli, and Dorigo, 2001; Blum and Dorigo, 2004) in the context of combinatorial problems with binary coded solutions, is to normalize the quality function, hence obtaining an automatic scaling of the pheromone values:

$$\tau_i \leftarrow (1-\rho)\tau_i + \rho\frac{\sum_{s\in S_t} Q_f(s)s_i}{\sum_{s\in S_t} Q_f(s)}. \tag{12}$$

While all the updates described above are of a somewhat heuristic nature, the stochastic gradient ascent and the cross-entropy methods allow to derive parameters update rules in a more systematic manner, as we show in the next two subsections.

3.3. The stochastic gradient ascent update

In section 2.1 an update rule for the stochastic gradient was derived:

$$\mathcal{T}^{t+1} = \mathcal{T}^t + \alpha_t \sum_{s \in S_t} Q_f(s) \nabla \ln P_{\mathcal{T}^t}(s), \tag{13}$$

where S_t is the sample at stage t.

As shown in Meuleau and Dorigo (2002), in case the distribution is implicitly defined by an ACO-type construction process, parameterized by the vector of the pheromone values, $\mathcal{T}$, the gradient $\nabla \ln P_{\mathcal{T}}(s)$ can be efficiently calculated. The following calculation is a generalization of the one in Meuleau and Dorigo (2002).

From the definition of ANT_SOLUTION_CONSTRUCTION, it follows that, for $s = \langle c_1, c_2, \ldots \rangle$,

$$P_{\mathcal{T}}(s) = \prod_{k=1}^{|s|-1} P_{\mathcal{T}}\bigl(c_{k+1} | \text{pref}_k(s)\bigr), \tag{14}$$

where $\text{pref}_k(s)$ is the k-prefix of s, and consequently

$$\nabla \ln P_{\mathcal{T}}(s) = \sum_{k=1}^{|s|-1} \nabla \ln P_{\mathcal{T}}\bigl(c_{k+1} | \text{pref}_k(s)\bigr). \tag{15}$$

Finally, given a pair of components $(i, j) \in \mathcal{C}^2$, using equation (10) and assuming differentiability of F, it is easy to verify that:

– if $i = c_k$ and $j = c_{k+1}$ then

$$\begin{aligned}
\frac{\partial}{\partial \tau(i,j)} \bigl\{\ln P_{\mathcal{T}}\bigl(c_{k+1} | \text{pref}_k(s)\bigr)\bigr\} &= \frac{\partial}{\partial \tau(i,j)} \left\{ \ln \frac{F(\tau(i,j))}{\sum_{(i,y) \in J_{x_k}} F(\tau(i,y))} \right\} \\
&= \frac{\partial}{\partial \tau(i,j)} \left\{ \ln F\bigl(\tau(i,j)\bigr) - \ln \sum_{(i,y) \in J_{x_k}} F\bigl(\tau(i,y)\bigr) \right\} \\
&= \frac{F'(\tau(i,j))}{F(\tau(i,j))} - \frac{F'(\tau(i,j))}{\sum_{(i,y) \in J_{x_k}} F(\tau(i,y))} \\
&= \left\{ 1 - \frac{F(\tau(i,j))}{\sum_{(i,y) \in J_{x_k}} F(\tau(i,y))} \right\} \frac{F'(\tau(i,j))}{F(\tau(i,j))} \\
&= \bigl\{1 - P_{\mathcal{T}}\bigl(j | \text{pref}_k(s)\bigr)\bigr\} G\bigl(\tau(i,j)\bigr),
\end{aligned}$$

where $G(\cdot) = F'(\cdot)/F(\cdot)$ and the subscript of F was omitted for the clarity of presentation.

– if $i = c_k$ and $j \neq c_{k+1}$ then (by a similar argument)

$$\frac{\partial \ln(P_{\mathcal{T}}(c_{k+1}|\text{pref}_k(s)))}{\partial \tau(i,j)} = -P_{\mathcal{T}}\big(j|\text{pref}_k(s)\big)G\big(\tau(i,j)\big),$$

– if $i \neq c_k$, then $P_{\mathcal{T}}(c_{k+1}|\text{pref}_k(s))$ is independent of $\tau(i,j)$ and

$$\frac{\partial \ln(P_{\mathcal{T}}(c_{k+1}|\text{pref}_k(s)))}{\partial \tau(i,j)} = 0.$$

By combining these results, the following pheromone update rule is derived:

SGA_UPDATE

– $\forall s \in S_t, \forall (i,j) \in s\colon \tau(i,j) \leftarrow \tau(i,j) + \alpha_t Q_f(s) G(\tau(i,j))$,
– $\forall s = \langle c_1, \ldots, c_k, \ldots \rangle \in S_t, \forall i = c_k$, with $1 \leqslant k < |s|,\ \forall j\colon \tau(i,j)$
$\leftarrow \tau(i,j) - \alpha_t Q_f(s) P_{\mathcal{T}}(j|\text{pref}_k(s)) G(\tau(i,j))$.

Hence, any connection (i,j) used in the construction of a solution is reinforced by an amount $\alpha_t Q_f(s) G(\tau(i,j))$, and any connection *considered* during the construction has its pheromone values evaporated by an amount $\alpha_t Q_f(s) P_{\mathcal{T}}\big(j|\text{pref}_k(s)\big)G\big(\tau(i,j)\big)$. Note that, if the solutions are allowed to contain loops, a connection may be updated more than once for the same solution.

In order to guarantee stability of the resulting algorithm, it is desirable to have a bounded gradient $\nabla \ln P_{\mathcal{T}}(s)$. This means that a function F, for which $G = F'/F$ is bounded, should be used. Meuleau and Dorigo (2002) suggest using $F(\cdot) = \exp(\cdot)$, which leads to $G \equiv 1$. It should be further noted that if, in addition, $Q_f = 1/f$ and $\alpha_t = 1$, the reinforcement part becomes $1/f$ as in the original Ant System (Dorigo, Maniezzo, and Colorni, 1996).

3.4. The cross-entropy update

As we have shown in section 2.2, the CE approach requires solving the following intermediate problem:

$$P_{t+1} = \underset{P \in \mathcal{M}}{\text{argmax}} \sum_{s \in S_t} Q_f(s) \ln P(s). \tag{16}$$

Let us now consider this problem in more details in case of an ACO-type probabilistic model.

Since at the maximum the gradient must be zero, we have:

$$\sum_{s \in S_t} Q_f(s) \nabla \ln P_{\mathcal{T}}(s) = 0. \tag{17}$$

In some relatively simple cases, for example when the solution s is represented by an unconstrained string of bits of length n, $(s_1, \ldots, s_n)$, and there is a single parameter τ_i

for the ith position in the string, such that $P_{\mathcal{T}}(s) = \prod_i p_{\tau_i}(s_i)$, the equation system (17) reduces to a set of independent equations:

$$\frac{\mathrm{d}\ln p_{\tau_i}}{\mathrm{d}\tau_i} \sum_{\substack{s\in S_t\\ s_i=1}} Q_f(s) = -\frac{\mathrm{d}\ln(1-p_{\tau_i})}{\mathrm{d}\tau_i} \sum_{\substack{s\in S_t\\ s_i=0}} Q_f(s), \quad i = 1,\ldots,n \tag{18}$$

which may often be solved analytically. For example, for $p_{\tau_i} = \tau_i$ it can be verified that the solution of equation (18) is simply

$$p_{\tau_i} = \tau_i = \frac{\sum_{s\in S_t} Q_f(s)s_i}{\sum_{s\in S_t} Q_f(s)} \tag{19}$$

and, in fact, a similar solution also applies to a more general class of Markov chain models (Rubinstein, 2001).

Now, since the pheromone trails τ_i in equation (19) are random variables, whose values depend on the particular sample, we may wish to make our algorithm more robust by introducing some conservatism into the update. For example, rather than discarding the old pheromone values, the new values may be taken to be a convex combination of the old values and the solution of equation (19):

$$\tau_i \leftarrow (1-\rho)\tau_i + \rho\,\frac{\sum_{s\in S_t} Q_f(s)s_i}{\sum_{s\in S_t} Q_f(s)}. \tag{20}$$

The resulting update is identical to the one used in the hyper-cube framework for ACO (Blum, Roli, and Dorigo, 2001; Blum and Dorigo, 2004).

However, for many cases of interest, equations (17) are coupled and an analytical solution is unavailable. Nevertheless, in the actual implementations of the CE method the update was of the form of equation (19) (with some brief remarks about using equation (20)) (Rubinstein, 2001), which may be considered as an approximation to the exact solution of the cross-entropy minimization problem given by equation (8).

Since, in general, the exact solution is not available, an iterative scheme such as gradient ascent could be employed, as described in section 2.2. As we have shown in the previous section, the gradient of the log-probability may be calculated as follows:

- if $i = c_k$ and $j = c_{k+1}$ then
$$\frac{\partial \ln(P_{\mathcal{T}}(c_{k+1}|\mathrm{pref}_k(s)))}{\partial\tau(i,j)} = \big(1 - P_{\mathcal{T}}\big(j|\mathrm{pref}_k(s)\big)\big)G\big(\tau(i,j)\big),$$
- if $i = c_k$ and $j \neq c_{k+1}$ then
$$\frac{\partial \ln(P_{\mathcal{T}}(c_{k+1}|\mathrm{pref}_k(s)))}{\partial\tau(i,j)} = -P_{\mathcal{T}}\big(j|\mathrm{pref}_k(s)\big)G\big(\tau(i,j)\big),$$
- if $i \neq c_k$ then
$$\frac{\partial \ln(P_{\mathcal{T}}(c_{k+1}|\mathrm{pref}_k(s)))}{\partial\tau(i,j)} = 0.$$

and these values may be plugged into any general iterative solution scheme of the cross-entropy minimization problem, for example, the one described by equation (9).

To conclude, we have shown that if we use (19) as a (possibly approximate) solution of equation (8), the same pheromone update rule as in the hyper-cube framework for ACO is derived. If otherwise we use a single-step gradient ascent for solving the problem given by equation (8), we obtain a generalization of the SGA update, in which the quality function is allowed to change over time.

4. Model-based genetic algorithms

In the “pure” model-based search, as it was described in the introduction, the parameterized model is iteratively updated, using the information extracted from the sample. However, if the whole search history is compressed into a single vector of the model’s parameters, a lot of useful information may be lost. In order to make a better use of the previous samples, many existing MBS algorithms use an auxiliary memory, in which they store some additional information collected during the search. This information is then used together with the latest sample for updating the model. For example, as we have seen in section 3.2, some existing ACO algorithms store the cost of the best-so-far solution or the average of the costs of the recent solutions. Another alternative would be to store several high-quality solutions encountered during the search. This is exactly what is being done in the majority of *estimation of distribution algorithms* (EDAs), recently developed within the evolutionary computation community.

In the following we give a brief overview of some existing EDAs and discuss their relations to the MBS algorithms described in the previous sections.

4.1. Estimation of distribution algorithms

As already mentioned in section 1, the classical genetic algorithm (GA) can be considered to be an example of the instance-based approach, in which the search is carried out by evolving the population of candidate solutions (typically represented by a string of bits) using selection, crossover and mutation operators (Holland, 1975).

The classical GA approach relies heavily on the assumption that there are some *building blocks* from which a good solution can be constructed. Moreover, it is assumed that with a proper choice of the crossover operator, these blocks will be (implicitly) detected and maintained in the population, while the selection operator will bias the search toward low-cost solutions. However, in practice, finding an appropriate crossover operator turns out to be a difficult task, while using some “general purpose” crossover operators often leads to poor performance. Another problem is the existence of *genetic drift* (Goldberg and Segrest, 1987), that is, a loss of population diversity due to the finite population size, and, as a result, a premature convergence to sub-optimal solutions.

In order to cope with the finite-population effects and also as an attempt to find an efficient alternative to the crossover/mutation operators, the estimation of distribution algorithms (Mühlenbein, Bendisch, and Voigt, 1996) were proposed. These algorithms

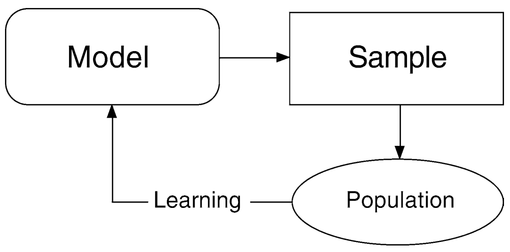

Figure 3. Graphic description of the estimation of distribution algorithms.

generate new solutions using probabilistic models, instead of crossover and mutation, and may be described using the following generic scheme:

```
EDA_ITERATION
  – Generate new solutions using the current probabilistic model.
  – Replace (some of) the old solutions by the new ones.
  – Modify the model using the new population.
```

This scheme, which may be seen as a particular type of MBS with auxiliary memory, is represented graphically in figure 3.

Different EDAs use different methods for construction/modification of the probabilistic model. However, most of them use the same method for estimating model parameters – a (possibly weighted) maximum-likelihood estimation. In this respect they are all closely related to the cross-entropy method described earlier and, as we show in the following, some of them employ particular forms of CE-type update.

In the remainder of this section we give an overview of existing estimation of distribution algorithms and discuss their relations with the algorithms presented in the previous sections of the paper. We consider two major classes of EDAs. The first class contains the algorithms that use a fixed simple model, which assumes that there are no interactions between the different string positions, that is, that the assignments to the different positions are independent. We observe that this is a particular kind of ACO-type model and show that all these algorithms lead to particular forms of ACO-type updates. The algorithms in the second class allow for dependencies between the positions, and, consequently, try to infer both the model structure and the model's parameters. Unlike the first group, both the models and the update mechanisms used by the algorithms in the second group are different from the ones used in the ACO framework.

It should be noted that all of the following algorithms were originally formulated for maximization problems, hence the obvious changes were done in order to translate them into the minimization setting that we consider in this paper.

4.1.1. Assuming independence between string positions

All the algorithms presented in this section create the new solutions, coded as binary vectors, by independently generating assignments for every position, with the ith position having probability p_i to take value 1. This may be considered a particularly simple ACO-type model, in which the components correspond to bit assignments, pheromone trails are associated with components, and there are no constraints.

The idea was initially proposed in Syswerda (1993), where the necessary probabilities were calculated as weighted frequencies over the population and randomly perturbed in order to simulate mutation. Apart from the mutation component, which seems to be an historical artifact borrowed from the classical GA and absent in later algorithms, this method is clearly an instance of the MBS with auxiliary memory in the form of the solution population, which uses the HCF-ACO-type (or, equivalently, CE-type) update with learning rate $\rho = 1$ for constructing the probabilistic model.

A similar approach was used in the *univariate marginal distribution algorithm (UMDA)* (Mühlenbein, Bendisch, and Voigt, 1996), the only difference being that in UMDA explicit classical selection procedures were used instead of giving weights to the solutions.

This idea is pushed even further in the population-based incremental learning (PBIL) algorithm (Baluja, 1994; Baluja and Caruana, 1995), where the population is completely replaced by a probability vector,[13] $\bar{p}$, with all p_i's initially set to 0.5. At every iteration a sample S is generated using the probability vector and then the probability vector is updated as follows:

```
PBIL_UPDATE
  – $S_{best} \leftarrow$ a fixed number of lowest cost solutions from $S$,
  – for every $s \in S_{best}$
        $p_i \leftarrow (1 - \rho)p_i + \rho s_i$,
```

where ρ is the learning rate.

As it can be easily seen, this update is virtually identical to the HCF-ACO update with the quality function being the indicator for the lowest cost solutions. In particular, if only the best solution is used for the update, HCF-ACO with iteration-best update is obtained.

Finally, the *compact genetic algorithm* (cGA) (Harik, Lobo, and Goldberg, 1999) was proposed as a modification of PBIL, intended to represent more faithfully the dynamics of the real genetic algorithm. Specifically, cGA simulates a genetic algorithm, with population size n and steady-state binary tournament selection, in the following way. At every iteration two solutions, a and b, are generated using the probability vector, and then the probability vector is updated as follows (assuming, without loss of generality, that a has lower cost):

```
cGA_UPDATE
  – if $a_i \neq b_i$ then
        if $a_i = 1$ then $p_i \leftarrow p_i + 1/n$,
                   else $p_i \leftarrow p_i - 1/n$.
```

This basic scheme can be extended to larger samples. Two variants were proposed in (Harik, Lobo, and Goldberg, 1999). In the first variant, intended to simulate tournaments of size m, a sample S of size m is generated and the basic update above is used for every pair in the set $\{(s^{best}, b) \mid b \in S, b \neq s^{best}\}$. In the second variant, a "round-robin tournament" is simulated, that is, the basic update is used for every pair of solutions from the sample.

Note that the basic cGA update can also be written in ACO-like form as:

$$p_i \leftarrow p_i + \frac{1}{n}(a_i - b_i). \tag{21}$$

Consequently, it can be shown that the update for "tournament of size m" cGA can be written as:

$$p_i \leftarrow p_i + \rho \sum_{s \in S} Q(s) s_i - \frac{\rho}{m} \sum_{s \in S} s_i, \tag{22}$$

where $\rho = m/n$ and

$$Q(s) = \begin{cases} 1 & \text{if } s = s^{best}, \\ 0 & \text{otherwise.} \end{cases} \tag{23}$$

For the "round-robin tournament" cGA, it can be shown that the update can also be described by equation (22), with $\rho = m(m+1)/n$ and

$$Q(s) = \frac{2 \cdot \text{rank}(s)}{m(m+1)}, \tag{24}$$

where the highest rank, m, is assigned to s^{best}.

It can be easily verified that these two updates are virtually identical to the HCF-ACO iteration-best and rank-based updates, respectively. The only difference between cGA and HCF-ACO is in the form of the evaporation factor. In cGA it is equal to $\frac{\rho}{m} \sum_{s \in S} s_i$, whereas in HCF-ACO it is equal to ρp_i, which is simply the expected value of the former.

4.1.2. Modeling dependencies between string positions

All the algorithms described in section 4.1.1 assumed a fixed model for the solutions' distribution, namely independence between assignments at different positions, and proposed different rules for calculating the parameters of the model. However, it may well happen that certain components produce good solutions only in conjunction with others, hence there may be strong dependencies within the population distribution.

Once the algorithm tries to model these *a priori* unknown dependencies between the solution constituents, the simple fixed structure has to be abandoned and the correct structure needs to be inferred together with the model's parameters.[14]

In the first EDAs that abandoned the independence assumption, only pairwise interactions were covered. The *mutual-information-maximizing input clustering (MIMIC)* algorithm (De Bonet, Isbell, and Viola, 1997), which was already mentioned earlier in the context of the cross-entropy method, maintains a population of the best solutions seen so far and constructs a chain distribution as a model of population by minimizing the Kullback–Leibler divergence between the model and the population distribution. Since finding the optimal chain distribution is an NP-hard problem, MIMIC uses a greedy search procedure for constructing the chain. For a given structure, the conditional probabilities (which are the parameters of the model) are estimated using the sample frequencies.

Baluja and Davies (1997) extend MIMIC in two important respects. First, they use a broader class of dependency trees instead of chain distributions, and, consequently, they are able to present an exact polynomial algorithm, rather than a greedy approximation. Second, instead of explicitly storing the population, the algorithm's history is summarized in a matrix of pairwise joint frequencies (with more weight given to recent instances), which are later used for optimal tree construction.

A somewhat more heuristic approach is taken in the *bivariate marginal distribution algorithm* (Pelikan and Mühlenbein, 1999), where the population is modeled using a forest, that is, a set of mutually independent dependency trees.[15] The model structure is determined using a Pearson's χ-square test (Marascuilo and McSweeney, 1977) for detecting dependencies.

The attempt to obtain yet more general models led to two different approaches. The first, the *extended compact genetic algorithm* (Harik, 1999), is a brute-force generalization of the UMDA, with the population modeled using a marginal product model. In the marginal product model the variables are divided into a number of independent clusters, while within a cluster any distribution is permitted. The cluster structure is determined by greedily optimizing the minimum description length metric (Mitchell, 1997) and the inter-cluster distributions are estimated using the population frequencies. The second approach, which is a generalization of ideas behind the tree-based algorithms described earlier, is to use a Bayesian network for modeling the population (Pelikan, Goldberg, and Cantú-Paz, 1998; Etxeberria and Larrañaga, 1999), with the network structure determined using some standard techniques for Bayesian network learning (Heckerman, 1995).

To summarize, all the algorithms described in this section use probabilistic models that are different from the one employed in ACO. Various criteria are used for choosing the model structure, but in all these algorithms a (weighted) maximum-likelihood (or, equivalently, minimal cross-entropy) method is used for estimating the model's parameters.

5. Conclusions

During the last decade a new approach for solving combinatorial optimization problems has been emerging. This approach, which we refer to as model-based search (MBS), tackles the combinatorial problem by sampling the solution space using a probabilistic model, which is adaptively modified as the search proceeds.

We observe that any successful algorithm belonging to the MBS framework is characterized by two components: a probabilistic model, which should allow an efficient generation of the candidate solutions, and a model update rule, which allows to concentrate the sampling in the high-quality regions. Accordingly, we describe two general approaches, the stochastic gradient ascent and the cross-entropy methods, for updating the model's parameters and we observe some previously unknown relationships between the two methods. Further, we demonstrate how the stochastic gradient ascent and the cross-entropy methods can be applied in the context of ant colony optimization which is a typical representative of the MBS approach. Moreover, we also show that in some cases the resulting updates coincide with existing ACO updates. Finally, we show that estimation of distribution algorithms, proposed in the field of genetic algorithms, also fall into the MBS framework, and that they are closely related to the other algorithms considered in this paper.

While sharing a lot of similar traits, each of the methods considered in this paper has some distinctive characteristics. Consequently, many interesting questions arise as to whether these peculiarities are contributing to the algorithm's performance.

For example, some of the estimation of distribution algorithms, which are the subject of section 4, contain at least one of the two following important components, absent in other approaches considered in this paper. The first is a population of solutions, which evolves throughout the search process and is used for constructing the probabilistic model. The other is the use of a flexible model structure, which is determined using an appropriate learning algorithm. However, it is still unclear whether either of these components gives any advantage in solving real-life problems. In addition, to the best of our knowledge, all the dependency-learning EDAs described in section 4.1.2 have been applied only to unconstrained optimization problems, which is a rather atypical situation in combinatorial optimization.[16] It remains to be seen whether similar algorithms can be designed for a more general setting. It should be further noted that, if a flexible model structure is shown to be beneficial in model-based search, some new model-selection rules should probably be used. The use of general purpose model-selection rules, borrowed from the machine learning field, seems to be inappropriate in the optimization context, since complex models are usually computationally more expensive, hence a stronger (than in generic learning) bias toward simpler models should probably be imposed.

Another interesting research direction, suggested by the approach presented in Baluja and Davies (1997), is to use a collection of sufficient statistics rather than a population, for the construction of the probabilistic model. This can be seen as a kind of two-stage learning procedure, where the statistics are learned incrementally, in a manner similar to ACO, but the actual (second-stage) model is re-constructed in every iteration using the first-stage statistics instead of raw samples.

Finally, the choice of the quality function, which provides a link between the original cost function and the model update rule, clearly has a crucial effect on the algorithms' dynamics. Some of the algorithms described in this paper use iteration-independent quality functions, while others adapt the quality function based on the search history. However, the issue of appropriate quality function choice is still poorly understood and is clearly an interesting future research direction.

Evaluating the utility of the different characteristics of the MBS algorithms clearly requires a serious experimental work. A first step in this direction was made in Zlochin and Dorigo (2002), where several MBS algorithms were rigorously compared over a class of MAX-SAT problems.

To conclude, considering all these algorithms within a common general framework provides a better understanding of what are the important parts of the algorithm and what is just an historical artifact due to a particular background of its proponents. Hopefully, the results presented in this paper will facilitate cross-fertilization between the considered model-based search methods and, perhaps, provide useful guidelines for designing new efficient optimization algorithms.

Acknowledgments

Mark Zlochin acknowledges the support of a Training Site fellowship funded by the Improving Human Potential (IHP) programme of the Commission of the European Community (CEC), grant HPRN-CT-2000-00032. Mauro Birattari acknowledges the support of a fellowship granted by the "Metaheuristics Network," a Research Training Network funded by the Improving Human Potential programme of the CEC, grant HPRN-CT-1999-00106. Nicolas Meuleau acknowledges the support of a Marie Curie fellowship funded by the CEC, grant HPMF-CT-2000-00495. Marco Dorigo acknowledges the support of the Belgian FNRS, of which he is a Research Director. More generally, this work was partially supported by the "Metaheuristics Network," a Research Training Network funded by the Improving Human Potential programme of the CEC, grant HPRN-CT-1999-00106. The information provided in this paper is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication.

Notes

1. There is, however, a rather close connection between these two usages of the term "model," as the model adaptation in combinatorial optimization may be considered as an attempt to model (in the reinforcement learning sense) the structure of the "promising" solutions.
2. The obvious changes must be done if a maximization problem is considered.
3. The model's structure may be fixed in advance, with solely the model's parameters being updated, or alternatively, the structure of the model may be allowed to change as well.
4. Technically, the smoothness assumption means that the function is continuously differentiable.
5. For the treatment of further details, we refer the interested reader to the original works of Rubinstein (1999a).
6. Similar results can be shown for many time-dependent quality functions.
7. As a simple example, consider the case where $\mathcal{M}$ contains all distributions over the binary variables x, y such that x and y are independent, and the quality function is $Q(x, y) = 2$, if $x = y = 0$, and 1 otherwise. If, for example, P_0 is the uniform distribution (hence in $\mathcal{M}$), then $\widehat{P}(x, y) = \frac{2}{5}$, if $x = y = 0$, and $\frac{1}{5}$ otherwise, and it can be easily verified that $\widehat{P}_1$ is not in $\mathcal{M}$.
8. This kind of quality function was also used in (De Bonet, Isbell, and Viola, 1997).

9. It should be noted that the same type of model was later (although independently) used in the CE framework under the name "associated stochastic network" (Rubinstein, 1999a, 2001).
10. How this mapping can be done in practice has been described in a number of earlier papers on the ACO metaheuristic; see, for example, Dorigo and Di Caro (1999), Dorigo, Di Caro, and Gambardella (1999).
11. Pheromone trails can be associated to components, connections, or both. In the following, unless stated otherwise, we assume that the pheromone trails are associated to connections, so that $\tau(i, j)$ is the pheromone associated to the connection between components i and j. It is straightforward to extend the algorithms to the other cases.
12. This situation may be prevented by allowing artificial ants to build infeasible solutions as well. In such a case an infeasibility penalty term is usually added to the cost function. However, it should be noted that in most settings ACO was applied to, the dead-end situation does not occur.
13. In this sense PBIL belongs to the MBS approach in its "pure" form and is, in fact, one of the first published algorithms belonging to the MBS approach.
14. Note, however, that in ACO models, pairwise dependencies may be learned implicitly, when the pheromone trails are associated with the connections between the components. Hence ACO provides an alternative way of learning pairwise dependencies, while still maintaining a fixed-structure model.
15. While seemingly more general, this class is in fact equivalent to the class of dependency trees, as any forest can be represented using a tree with degenerate links.
16. Although for some problems sophisticated schemes for coding the solutions as unconstrained binary strings have been devised (e.g., see (Baluja, 1994)), all the useful dependencies between the solution components may be hidden by these coding schemes.

References

Aarts, E.H.L. and J.K. Lenstra. (1997). *Local Search in Combinatorial Optimization*. Chichester, UK: Wiley.

Baluja, S. (1994). "Population-Based Incremental Learning: A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning." Technical Report CMU-CS-94-163, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Baluja, S. and R. Caruana. (1995). "Removing the Genetics from the Standard Genetic Algorithm." In *Proceedings of the Twelfth International Conference on Machine Learning (ML-95)*. Palo Alto, CA: Morgan Kaufmann, pp. 38–46.

Baluja, S. and S. Davies. (1997). "Using Optimal Dependency-Trees for Combinatorial Optimization: Learning the Structure of the Search Space." In *Proceedings of the Fourteenth International Conference on Machine Learning (ML-97)*. Palo Alto, CA: Morgan Kaufmann, pp. 30–38.

Bertsekas, D.P. (1995). *Nonlinear Programming*. Belmont, MA: Athena Scientific.

Blum, C., A. Roli, and M. Dorigo. (2001). "HC-ACO: The Hyper-Cube Framework for Ant Colony Optimization." In *Proceedings of MIC'2001 – Meta-Heuristics International Conference*, Vol. 2, Porto, Portugal, pp. 399–403.

Blum, C. and M. Dorigo. (2004). "The Hyper-Cube Framework for Ant Colony Optimization." *IEEE Transactions on Systems, Man, and Cybernetics, Part B* 34(2), 1161–1172.

De Boer, P.T., D.P. Kroese, S. Mannor, and R.Y. Rubinstein. (2001). "A Tutorial on the Cross-Entropy Method." `http://wwwhome.cs.utwente.nl/~ptdeboer/ce/tutorial.pdf`

De Bonet, J.S., C.L. Isbell, and P. Viola. (1997). "MIMIC: Finding Optima by Estimating Probability Densities." In *Advances in Neural Information Processing Systems*, Vol. 9. Cambridge, MA: MIT Press, pp. 424–431.

Dorigo, M. (1992). "Optimization, Learning and Natural Algorithms," Ph.D. Thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy (in Italian).

Dorigo, M. and G. Di Caro. (1999). "The Ant Colony Optimization Meta-Heuristic." In *New Ideas in Optimization*. London, UK: McGraw Hill, pp. 11–32.

Dorigo, M., G. Di Caro, and L.M. Gambardella. (1999). "Ant Algorithms for Discrete Optimization." *Artificial Life* 5(2), 137–172.

Dorigo, M. and L.M. Gambardella. (1997). "Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem." *IEEE Transactions on Evolutionary Computation* 1(1), 53–66.

Dorigo, M., V. Maniezzo, and A. Colorni. (1991). "The Ant System: An Autocatalytic Optimizing Process." Technical Report 91-016 Revised, Dipartimento di Elettronica, Politecnico di Milano, Italy.

Dorigo, M., V. Maniezzo, and A. Colorni. (1996). "Ant System: Optimization by a Colony of Cooperating Agents." *IEEE Transactions on Systems, Man, and Cybernetics – Part B* 26(1), 29–41.

Dorigo, M. and T. Stützle. (2002). "The Ant Colony Optimization Metaheuristic: Algorithms, Applications and Advances." In *Handbook of Metaheuristics*, International Series in Operations Research & Management Science, Vol. 57. Norwell, MA: Kluwer Academic, pp. 251–285.

Dorigo, M. and T. Stützle. (2004). *Ant Colony Optimization*. Cambridge, MA: MIT Press.

Etxeberria, R. and P. Larrañaga. (1999). "Global Optimization with Bayesian Networks." In *Proceedings of the Second Symposium on Artificial Intelligence*, La Habana, Cuba, pp. 332–339.

Goldberg, D. and P. Segrest. (1987). "Finite Markov Chain Analysis of Genetic Algorithms." In *Proceedings of the Second International Conference on Genetic Algorithms*. Hillsdale, NJ: Lawrence Erlbaum Associates, pp. 1–8.

Harik, G.R. (1999). "Linkage Learning via Probabilistic Modeling in the ECGA." Technical Report IlliGAL, 99010, University of Illinois at Urbana-Champaign, Urbana, IL.

Harik, G.R., F.G. Lobo, and D.E. Goldberg. (1999). "The Compact Genetic Algorithm." *IEEE Transactions on Evolutionary Computation* 3(4), 287–297.

Heckerman, D. (1995). "A Tutorial on Learning with Bayesian Networks." Technical Report MSR-TR-95-06, Microsoft Research, Redmond, WA.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems.* Ann Arbor, MI: University of Michigan Press.

Kullback, S. (1959). *Information Theory and Statistics*. New York: Wiley.

Larrañaga, P. and J.A. Lozano. (2001). *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation*. Boston, MA: Kluwer Academic.

Lieber, D. (1999). "The Cross-Entropy Method for Estimating Probabilities of Rare Events." Ph.D. Thesis, William Davidson Faculty of Industrial Engineering and Management, Technion, Haifa, Israel.

Maniezzo, V. (1999). "Exact and Approximate Nondeterministic Tree-Search Procedures for the Quadratic Assignment Problem." *INFORMS Journal on Computing* 11(4), 358–369.

Marascuilo, L. and M. McSweeney. (1977). *Nonparametric and Distribution-Free Methods for the Social Sciences*. Monterey, CA: Brooks/Cole.

Meuleau, N. and M. Dorigo. (2002). "Ant Colony Optimization and Stochastic Gradient Descent." *Artificial Life* 8(2), 103–121.

Mitchell, T.M. (1997). *Machine Learning*. New-York: McGraw-Hill.

Mühlenbein, H., J. Bendisch, and H.-M. Voigt. (1996). "From Recombination of Genes to the Estimation of Distributions. I. Binary Parameters." In *Proceedings of PPSN-I, First International Conference on Parallel Problem Solving from Nature*. Berlin, Germany: Springer, pp. 178–187.

Pelikan, M., D.E. Goldberg, and E. Cantú-Paz. (1998). "Linkage Problem, Distribution Estimation, and Bayesian Networks." Technical Report IlliGAL, 98013, University of Illinois at Urbana-Champaign, Urbana, IL.

Pelikan, M., D.E. Goldberg, and F. Lobo. (1999). "A Survey of Optimization by Building and Using Probabilistic Models." Technical Report IlliGAL, 99018, University of Illinois at Urbana-Champaign, Urbana, IL.

Pelikan, M. and H. Mühlenbein. (1999). "The Bivariate Marginal Distribution Algorithm." In *Advances in Soft Computing – Engineering Design and Manufacturing*. London, UK: Springer, pp. 521–535.

Quinlan, J. (1993). "Combining Instance-Based and Model-Based Learning." In *Proceedings of the Tenth International Conference on Machine Learning (ML-93)*. San Mateo, CA: Morgan Kaufmann, pp. 236–243.

Robbins, H. and S. Monro. (1951). "A Stochastic Approximation Method." *Annals of Mathematical Statistics* 22, 400–407.

Rubinstein, R.Y.: 1999a, "The Cross-Entropy Method for Combinatorial and Continuous Optimization." *Methodology and Computing in Applied Probability* 1(2), 127–190.

Rubinstein, R.Y.: 1999b, "Rare Event Simulation via Cross-Entropy and Importance Sampling." In *Second International Workshop on Rare Event Simulation, RESIM'99*, pp. 1–17.

Rubinstein, R.Y. (2001). "Combinatorial Optimization, Cross-Entropy, Ants and Rare Events." In *Stochastic Optimization: Algorithms and Applications*. Dordrecht, The Netherlands: Kluwer Academic, pp. 303–364.

Stützle, T. and H.H. Hoos. (1997). "The MAX-MIN Ant System and Local Search for the Traveling Salesman Problem." In *Proceedings of ICEC'97 – 1997 IEEE 4th International Conference on Evolutionary Computation*. Piscataway, NJ: IEEE Press, pp. 308–313.

Stützle, T. and H.H. Hoos. (2000). "$\mathcal{MAX}$–$\mathcal{MIN}$ Ant System." *Future Generation Computer Systems* 16(8), 889–914.

Sutton, R. and A. Barto. (1998). *Reinforcement Learning. An Introduction*. Cambridge, MA: MIT Press.

Syswerda, G. (1993). "Simulated Crossover in Genetic Algorithms." In *Foundations of Genetic Algorithms*, Vol. 2. San Mateo, CA: Morgan Kaufmann, pp. 239–255.

Zlochin, M. and M. Dorigo. (2002). "Model-Based Search for Combinatorial Optimization: A Comparative Study." In *Proceedings of PPSN-VII, Seventh International Conference on Parallel Problem Solving from Nature*, Lecture Notes in Computer Science, Vol. 2439. Berlin, Germany: Springer, pp. 651–661.