Deception in Ant Colony Optimization

Christian Blum and Marco Dorigo

IRIDIA – Université Libre de Bruxelles – Brussels, Belgium
{cblum,mdorigo}@ulb.ac.be

Abstract. The search process of a metaheuristic is sometimes misled. This may be caused by features of the tackled problem instance, by features of the algorithm, or by the chosen solution representation. In the field of evolutionary computation, the first case is called *deception* and the second case is referred to as *bias*. In this work we formalize the notions of deception and bias for ant colony optimization. We formally define *first order deception* in ant colony optimization, which corresponds to deception as being described in evolutionary computation. Furthermore, we formally define *second order deception* in ant colony optimization, which corresponds to the bias introduced by components of the algorithm in evolutionary computation. We show by means of an example that second order deception is a potential problem in ant colony optimization algorithms.

1 Introduction

Deception and bias are well-studied subjects in evolutionary computation algorithms for the application to combinatorial optimization (CO) problems. The term deception was introduced by Goldberg in [12] with the aim of describing problems that are misleading for genetic algorithms (GAs). Well-known examples of deceptive problems are n-bit trap functions [7]. These functions are characterized by fixpoints with large basins of attraction that correspond to sub-optimal solutions, and by fixpoints with relatively small basins of attraction that correspond to optimal solutions. Therefore, for these problems a GA will – in most cases – not find an optimal solution. Except for deception, other efforts in the field of evolutionary computation were aimed at studying the effects of the bias that is sometimes introduced by the solution representation and the genetic operators. In some cases this bias was shown to have a negative impact on the search process (see, for example, [16]).

In the early 90's, ant colony optimization (ACO) [8–10] emerged as a novel nature-inspired metaheuristic method for the solution of combinatorial optimization problems. The inspiring source of ACO is the foraging behavior of real ants. When searching for food, ants initially explore the area surrounding their nest in a random manner. As soon as an ant finds a food source, it evaluates quantity and quality of the food and carries some of the food found to the nest. During the return trip, the ant deposits a chemical pheromone trail on the ground. The quantity of pheromone deposited, which may depend on the quantity and quality of the food, will guide other ants to the food source. The indirect communication between the ants via the pheromone trails allows them to find shortest paths between their nest and food sources. This behaviour of real ant colonies is exploited in artificial ant colonies in order to solve discrete optimization problems.

Research on bias in ACO algorithms is largely restricted to the work by Merkle and Middendorf [14, 13], and the work by Blum and Sampels [3, 4]. Merkle and Middendorf proposed and introduced the use of *models of ACO algorithms*. They studied the behaviour of a simple ACO algorithm by studying the dynamics of its model when applied to permutation problems. It was shown that the behaviour of ACO algorithms is strongly influenced by the pheromone model (and the way of using it) and by the competition between the ants. Moreover, it was shown that the performance of the model of an ACO algorithm may decrease during a run, which is clearly undesirable, because in general this worsens the probability of finding better and better solutions. Independently, Blum and Sampels showed on the example of an ACO algorithm for a general shop scheduling problem that ACO algorithms may fail depending on the chosen pheromone model.

Our Contribution. In this work we formalize the notions of deception and bias in ant colony optimization. We formally define *first order deception* in ant colony optimization, which corresponds to deception as being described in evolutionary computation. Then, we formally define *second order deception* in ant colony optimization, which corresponds to the bias introduced by components of the algorithm in evolutionary computation. Finally, we give an example of second order deception in order to show that the bias that leads to the occurrence of second order deception can make an ACO algorithm fail.

2 The Framework of a Basic ACO Algorithm

ACO algorithms are metaheuristic methods for tackling combinatorial optimization problems (see [11]). The central component of an ACO algorithm is the pheromone model, which is used to probabilistically sample the search space. As outlined in [2], the pheromone model can be derived from a *model* of the CO problem under consideration. A model of a CO problem can be stated as follows.

Definition 1. *A **model** $\mathcal{P} = (\mathcal{S}, \Omega, f)$ of a CO problem consists of:*

- *a **search (or solution) space** $\mathcal{S}$ defined over a finite set of discrete decision variables and a set Ω of **constraints** among the variables;*
- *an **objective function** $f : \mathcal{S} \rightarrow \mathbb{R}^+$ to be minimized.*

*The search space $\mathcal{S}$ is defined as follows: Given is a set of n **discrete variables** X_i with domains $D_i = \{v_i^1, \ldots, v_i^{|D_i|}\}$, $i = 1, \ldots, n$. A variable instantiation, that is, the assignment of a value v_i^j to a variable X_i, is denoted by $X_i = v_i^j$. A feasible solution $s \in \mathcal{S}$ is a complete assignment (i.e., an assignment in which each decision variable has a domain value assigned) that satisfies the constraints. If the set of constraints Ω is empty, then each decision variable can take any value*

Algorithm 1 The framework of a basic ACO algorithm

```
input: An instance $P$ of a CO problem model $\mathcal{P} = (\mathcal{S}, f, \Omega)$.
InitializePheromoneValues($\mathcal{T}$)
$\mathfrak{s}_{bs} \leftarrow$ NULL
while termination conditions not met do
  $\mathfrak{S}_{iter} \leftarrow \emptyset$
  for $j = 1, \ldots, n_a$ do
    $\mathfrak{s} \leftarrow$ ConstructSolution($\mathcal{T}$)
    $\mathfrak{s} \leftarrow$ LocalSearch($\mathfrak{s}$)        {optional}
    if ($f(\mathfrak{s}) < f(\mathfrak{s}_{bs})$) or ($\mathfrak{s}_{bs} =$ NULL) then $\mathfrak{s}_{bs} \leftarrow \mathfrak{s}$
    $\mathfrak{S}_{iter} \leftarrow \mathfrak{S}_{iter} \cup \{\mathfrak{s}\}$
  end for
  ApplyPheromoneUpdate($\mathcal{T}$,$\mathfrak{S}_{iter}$,$\mathfrak{s}_{bs}$)
end while
output: The best-so-far solution $\mathfrak{s}_{bs}$
```

from its domain independently of the values of the other decision variables. In this case we call $\mathcal{P}$ an ***unconstrained*** *problem model, otherwise a* ***constrained*** *problem model. A feasible solution $s^* \in \mathcal{S}$ is called a* ***globally optimal solution****, if $f(s^*) \leq f(s)$ $\forall s \in \mathcal{S}$. The set of globally optimal solutions is denoted by $\mathcal{S}^* \subseteq \mathcal{S}$. To solve a CO problem one has to find a solution $s^* \in \mathcal{S}^*$.*

A model of the CO problem under consideration implies the finite set of solution components and the pheromone model as follows. First, we call the combination of a decision variable X_i and one of its domain values v_i^j a *solution component* denoted by $\mathfrak{c}_i^j$. Then, the pheromone model consists of a *pheromone trail parameter* $\mathcal{T}_i^j$ for every solution component $\mathfrak{c}_i^j$. The value of a pheromone trail parameter $\mathcal{T}_i^j$ – called *pheromone value* – is denoted by τ_i^j. The set of all pheromone trail parameters is denoted by $\mathcal{T}$. As a CO problem can be modelled in different ways, different models of the CO problem can be used to define different pheromone models.

Algorithm 1 captures the framework of a basic ACO algorithm. It works as follows. At each iteration, n_a ants probabilistically construct solutions to the combinatorial optimization problem under consideration, exploiting a given pheromone model. Then, optionally, a local search procedure is applied to the constructed solutions. Finally, before the next iteration starts, some of the solutions are used for performing a pheromone update. The details of this framework are explained in more detail in the following.

InitializePheromoneValues($\mathcal{T}$). At the start of the algorithm the pheromone values are all initialized to a constant value $c > 0$.

ConstructSolution($\mathcal{T}$). The basic ingredient of any ACO algorithm is a constructive heuristic for probabilistically constructing solutions. A constructive heuristic assembles solutions as sequences of elements from the finite set of solution components $\mathfrak{C}$. A solution construction starts with an empty partial solution $\mathfrak{s}^p = \langle\rangle$. Then, at each construction step the current partial solution $\mathfrak{s}^p$ is extended by adding a feasible solution component from the set $\mathfrak{N}(\mathfrak{s}^p) \subseteq \mathfrak{C} \setminus \mathfrak{s}^p$, which is defined by the solution construction mechanism. The process of constructing solutions can be regarded as a walk (or a path) on the so-called *construction graph* $\mathcal{G}_C = (\mathfrak{C}, \mathfrak{L})$, which is a fully connected graph whose vertices are the solution components $\mathfrak{C}$ and the set $\mathfrak{L}$ are the connections. The allowed walks on $\mathcal{G}_C$ are implicitly defined by the solution construction mechanism that defines the set $\mathfrak{N}(\mathfrak{s}^p)$ with respect to a partial solution $\mathfrak{s}^p$. We denote the set of all solution that may be constructed in this way by $\mathfrak{S}$. The choice of a solution component from $\mathfrak{N}(\mathfrak{s}^p)$ is, at each construction step, done probabilistically. In most ACO algorithms the probabilities for choosing the next solution component – also called the *transition probabilities* – are defined as follows:

$$\mathbf{p}(\mathfrak{c}_i^j \mid \mathfrak{s}^p) = \frac{{\tau_i^j}^{\alpha} \cdot \eta(\mathfrak{c}_i^j)^{\beta}}{\sum\limits_{\mathfrak{c}_k^l \in \mathfrak{N}(\mathfrak{s}^p)} {\tau_k^l}^{\alpha} \cdot \eta(\mathfrak{c}_k^l)^{\beta}} \ , \ \ \forall\, \mathfrak{c}_i^j \in \mathfrak{N}(\mathfrak{s}^p) \ , \tag{1}$$

where η is a weighting function that assigns, at each construction step, a heuristic value $\eta(\mathfrak{c}_i^j)$ to each feasible solution component $\mathfrak{c}_i^j \in \mathfrak{N}(\mathfrak{s}^p)$. The values that are given by the weighting function are commonly called the *heuristic information.* α and β are positive parameters whose values determine the relative importance of pheromone and heuristic information.

ApplyPheromoneUpdate($\mathcal{T}$,$\mathfrak{S}_{iter}$,$\mathfrak{s}_{bs}$). Most ACO algorithms use the following pheromone value update rule:

$$\tau_i^j \leftarrow (1-\rho) \cdot \tau_i^j + \frac{\rho}{n_a} \cdot \sum_{\{\mathfrak{s} \in \mathfrak{S}_{upd} \mid \mathfrak{c}_i^j \in \mathfrak{s}\}} F(\mathfrak{s}) \ , \tag{2}$$

for $i = 1, \ldots, n$, and $j \in \{1, \ldots, |D_i|\}$ [1]. $\rho \in (0,1]$ is a parameter called *evaporation rate.* $F : \mathfrak{S} \mapsto \mathbb{R}^+$ is a function such that $f(\mathfrak{s}) < f(\mathfrak{s}') \Rightarrow F(\mathfrak{s}) \geq F(\mathfrak{s}')$, $\forall \mathfrak{s} \neq \mathfrak{s}' \in \mathfrak{S}$. $F(\cdot)$ is commonly called the *quality function.* Instantiations of this update rule are obtained by different specifications of $\mathfrak{S}_{upd}$, which – in all cases that we consider in this paper – is a subset of $\mathfrak{S}_{iter} \cup \{\mathfrak{s}_{bs}\}$, where $\mathfrak{S}_{iter}$ is the set of solutions that were constructed in the current iteration, and $\mathfrak{s}_{bs}$ is the best-so-far solution. A well-known example of update rule (2) is the AS-update rule (i.e., the update rule of Ant System (AS) [10]) which is obtained from (2) by setting $\mathfrak{S}_{upd} \leftarrow \mathfrak{S}_{iter}$. The goal of the pheromone value update rule is to increase the pheromone values on solution components that have been found in high quality solutions.

[1] Note that the factor $\frac{1}{n_a}$ is usually not used. We introduce it for the mathematical purpose of studying the expected update of the pheromone values. However, as the factor is constant it does not change the qualitative behaviour of an ACO algorithm.

3 Deception in Ant Colony Optimization

In the following, we first define and study first order deception in ACO, which corresponds to deception in EC. Then, we introduce second order deception in ACO, which corresponds to bias in EC.

3.1 First Order Deception

The desired behaviour of an ACO algorithm can be stated as follows: The average quality of the generated solutions should grow over time. This is desirable, as it usually increases the probability of finding better solutions over time. For studying the behaviour of an ACO algorithm, we study in the following its *model* as proposed by Merkle and Middendorf in [14]. The model of an ACO algorithm is obtained by applying the expected pheromone update instead of the real pheromone update. Therefore, models of ACO algorithms are deterministic and can be considered discrete dynamical systems. The behaviour of ACO algorithm models is characterized by the evolution of the expected quality of the solutions that are generated per iteration. We denote this *expected iteration quality* in the following by $W_F(\mathcal{T})$, or by $W_F(\mathcal{T} \mid t)$, where $t > 0$ is the iteration counter.

As an example, a simplified model of an ACO algorithm is obtained by assuming an infinite number of solution constructions (i.e., ants) per iteration. In this case, the expected iteration quality is given by

$$W_F(\mathcal{T}) = \sum_{\mathfrak{s} \in \mathfrak{S}} F(\mathfrak{s}) \cdot \mathbf{p}(\mathfrak{s} \mid \mathcal{T}) \ . \tag{3}$$

The expected pheromone update of the AS algorithm (i.e., Algorithm 1 using the AS-update rule) based on this simplified model can then be stated as follows:

$$\tau_i^j(t+1) \leftarrow (1-\rho) \cdot \tau_i^j(t) + \rho \cdot \sum_{\{\mathfrak{s} \in \mathfrak{S} \mid \mathfrak{c}_i^j \in \mathfrak{s}\}} F(\mathfrak{s}) \cdot \mathbf{p}(\mathfrak{s} \mid \mathcal{T}) \ , \tag{4}$$

for $i = 1, \ldots, n$, $j = 1, \ldots, |D_i|$ (note that j depends on i), and where t is the iteration counter. However, this simplified model is only an example, and more accurate models of ACO algorithms are possible (see for example [14]).

Definition 2. *Given a model $\mathcal{P}$ of a CO problem, we call a model of an ACO algorithm applied to any instance P of $\mathcal{P}$ a* **local optimizer** *if for any initial setting of the pheromone values the expected update of the pheromone values is such that $W_F(\mathcal{T} \mid t+1) \geq W_F(\mathcal{T} \mid t)$, $\forall t \geq 0$. In other words, the expected quality of the generated solutions per iteration must increase monotonically.*

Note that an increase in expected iteration quality does – due to the rather loose definition of the relation between quality function $F(\cdot)$ and objective function $f(\cdot)$ – not necessarily imply an increase in expected objective function values. However, in most cases this can be assumed. Based on this definition we introduce the definition of first order deceptive systems.

Definition 3. *Given a model $\mathcal{P}$ of a CO problem, we call a local optimizer* **applied to** *instance P of $\mathcal{P}$ a* **first order deceptive system (FODS)** *if there exists an initial setting of the pheromone values such that the algorithm does in expectation not converge to a globally optimal solution.*

This means that even if the model of an ACO algorithm is a local optimizer it is a first order deceptive system when for example applied to problem instances that are characterized by the fact that they induce more than one stable fixpoint of which at least one corresponds to a local minimum[2].

Ant System Applied to Unconstrained Problems. In [2] was proposed the hyper-cube framework (HCF) for ACO. The HCF is a framework for ACO algorithms that applies a normalized pheromone update at each iteration. For example, the AS-update rule in the HCF is

$$\tau_i^j \leftarrow (1-\rho)\cdot\tau_i^j + \rho\cdot \sum_{\{\mathfrak{s}\in\mathfrak{S}_{iter}\mid \mathfrak{c}_i^j\in\mathfrak{s}\}} \frac{F(\mathfrak{s})}{\sum_{\{\mathfrak{s}'\in\mathfrak{S}_{iter}\}} F(\mathfrak{s}')} \ , \tag{5}$$

for $i = 1,\ldots,n$, $j = 1,\ldots,|D_i|$. The difference between pheromone update rules in the HCF and the ones that are used in standard ACO algorithms consists in the normalization of the added amount of pheromone. In [2] it was shown that the simplified model of the AS algorithm in the HCF (i.e., assuming an infinite number of solution constructions per iteration) possesses the property that the expected iteration quality monotonically increases over time when applied to unconstrained problems. This can be regarded as an indicator that the average quality of the generated solutions in empirical applications (i.e., using a finite number of ants per iteration) is likely to grow from iteration to iteration. In this section we show that the simplified model of the AS algorithm (not implemented in the HCF) also possesses this property when applied to unconstrained problems, which can be stated as follows: given are n decision variables X_i, $i = 1,\ldots,n$, with domains $D_i = \{v_i^1,\ldots,v_i^{|D_i|}\}$. To construct a solution we do n construction steps as follows. At construction step i, where $i \in \{1,\ldots,n\}$, we add one of the solution components $\mathfrak{c}_i^j$, where $j \in \{1,\ldots,|D_i|\}$, to the current partial solution $\mathfrak{s}^p$ under construction. This corresponds to assigning a value to decision variable X_i. The transition probabilities are as follows:

$$\mathbf{p}(\mathfrak{c}_i^j \mid \mathcal{T}) = \frac{\tau_i^j}{\sum\limits_{k=1}^{|D_i|} \tau_i^k} \ , \quad i = 1,\ldots,n \ . \tag{6}$$

[2] The term of a stable fixpoint stems from the field of discrete dynamical systems. It denotes the state of a system that is characterized by the fact that when the system has entered this state, which is characterized by a an open basin of attraction, it will never leave it again.

Theorem 1. *The simplified model of AS (i.e., assuming an infinite number of solution constructions per iteration) is a local optimizer when applied to unconstrained problems.*

Proof. This theorem is a simple extension of Theorem 3 in [2], which proves that the simplified model of AS in the HCF is a local optimizer when applied to unconstrained problems. For distinguishing between the pheromone values of (a) AS and (b) AS in the HCF, we denote the pheromone values of AS by τ^a and the pheromone values of AS in the HCF by τ^b. We use the same notational convention for all the probabilities. Let us assume that at iteration $t \geq 0$ the pheromone values of AS and AS in the HCF are such that $\mathbf{p}^a(\mathfrak{c}_i^j \mid t) = \mathbf{p}^b(\mathfrak{c}_i^j \mid t)$, for $i = 1, \ldots, n$, and $j = 1, \ldots, |D_i|$. Assuming that $\rho = 1$, the expected update of the pheromone values of AS can, according to Equation 4, be stated as

$$\tau^{a}{}_i^j(t+1) \leftarrow \sum_{\{\mathfrak{s} \in \mathfrak{S} \mid \mathfrak{c}_i^j \in \mathfrak{s}\}} F(\mathfrak{s}) \cdot \mathbf{p}(\mathfrak{s} \mid \mathcal{T}) \ . \tag{7}$$

In the same way, according to Equation 5, the expected update of the pheromone values of AS in the HCF can be stated as

$$\tau^{b}{}_i^j(t+1) \leftarrow \sum_{\{\mathfrak{s} \in \mathfrak{S} \mid \mathfrak{c}_i^j \in \mathfrak{s}\}} \frac{F(\mathfrak{s}) \cdot \mathbf{p}(\mathfrak{s} \mid \mathcal{T})}{W_F(\mathcal{T})} \ . \tag{8}$$

Theorem 1 in [2] shows that $\mathbf{p}^b(\mathfrak{c}_i^j \mid t+1) = \tau^{b}{}_i^j(t+1)$, for $i = 1, \ldots, n$, and $j = 1, \ldots, |D_i|$. Therefore, with Equation 8 it holds that

$$\mathbf{p}^b(\mathfrak{c}_i^j \mid t+1) = \sum_{\{\mathfrak{s} \in \mathfrak{S} \mid \mathfrak{c}_i^j \in \mathfrak{s}\}} \frac{F(\mathfrak{s}) \cdot \mathbf{p}(\mathfrak{s} \mid \mathcal{T})}{W_F(\mathcal{T})} \ . \tag{9}$$

Furthermore, from Equations 6 and 7 it follows that

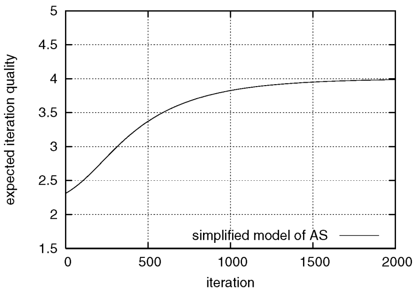
$$\mathbf{p}^a(\mathfrak{c}_i^j \mid t+1) = \frac{\tau^{a}{}_i^j(t+1)}{\sum\limits_{k=1}^{|D_i|} \tau^{a}{}_i^k(t+1)} = \sum_{\{\mathfrak{s} \in \mathfrak{S} \mid \mathfrak{c}_i^j \in \mathfrak{s}\}} \frac{F(\mathfrak{s}) \cdot \mathbf{p}(\mathfrak{s} \mid \mathcal{T})}{W_F(\mathcal{T})} \ . \tag{10}$$

From Equations 9 and 10 it follows that $\mathbf{p}^a(\mathfrak{c}_i^j \mid t+1) = \mathbf{p}^b(\mathfrak{c}_i^j \mid t+1)$, for $i = 1, \ldots, n$, and $j = 1, \ldots, |D_i|$. As the simplified model of AS in the HCF is a local optimizer as shown in Theorem 3 in [2], also the simplified model of AS is a local optimizer when $\rho = 1$. The general case follows from the fact that the new pheromone vector for $\rho < 1$ for AS in the HCF is on the line segment between the old pheromone vector and the pheromone vector that would be the result of the setting $\rho = 1$. The same holds for AS. □

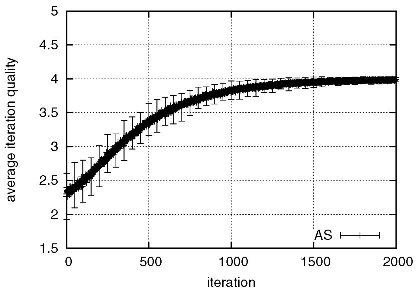
A consequence of this result is that the simplified model of AS is a FODS when applied to instances of unconstrained problems that induce more than one stable attractor such as for example n-bit trap functions. This is confirmed by

Table 1. A 4-bit trap function. Each of the 16 columns shows the 4 bits of a solution and its objective function value

1st bit	0	1	0	0	0	1	1	1	0	0	0	1	1	1	0	1
2nd bit	0	0	1	0	0	1	0	0	1	1	0	1	1	0	1	1
3rd bit	0	0	0	1	0	0	1	0	1	0	1	1	0	1	1	1
4th bit	0	0	0	0	1	0	0	1	0	1	1	0	1	1	1	1
$f(\cdot)$	5	1	1	1	1	2	2	2	2	2	2	3	3	3	3	4



(a) Simplified model of AS



(b) AS

Fig. 1. (a) Evolution of the expected iteration quality W_F of the simplified model of AS when applied to a 4-bit trap function as shown in Table 1. As in this case we are considering a maximization problem, we choose $F(\cdot) = f(\cdot)$. The expected iteration quality continuously increases and the system converges to the sub-optimal solution with quality 4 (the optimal solution is characterized by $X_i = 0$, $i = 1, \ldots, 4$, and quality 5). (b) Evolution of the average iteration quality of AS (i.e., Algorithm 1 using the AS-update rule) when applied to the 4-bit trap function. The graph is the result of 100 runs of the algorithms (the vertical bars show the standard deviation). For (a) and (b) we chose the following settings: $\rho = 0.01$ (evaporation rate), and $c = 0.5$ (initial setting for the pheromone values). Furthermore, for (b) we chose $n_a = 10$ (number of ants per iteration)

Figure 1 which shows the evolution of the expected quality W_F of the simplified model of AS when applied to a 4-bit trap function (see Table 1) over time. The result shows that ACO algorithms suffer from the same type of deception as EC algorithms.

3.2 Second Order Deception

In most cases, for example when NP-hard problems are considered, we can not do better than having a local optimizer. Therefore, first order deception can not be considered a problem. In contrast, the empirical behaviour of an algorithm whose model is not a local optimizer might be very difficult to predict. In the following, we define a second order deceptive system (SODS) as a system that does not have the property of being a local optimizer.

Definition 4. *Given a model $\mathcal{P}$ of a CO problem, we call a model of an ACO algorithm* **applied to** *instance P of $\mathcal{P}$ a* **second order deceptive system (SODS)***, if the evolution of the expected iteration quality contains time windows $[i, i+l]$ (where $i > 0$, $l > 0$) with $W_F(\mathcal{T} \mid t+1) < W_F(\mathcal{T} \mid t)$, $\forall\, t \in \{i, \ldots, i+l-1\}$. This means that the combination of a model of an ACO algorithm and an instance P of $\mathcal{P}$ may be a SODS, if the ACO algorithm model is not a local optimizer. We henceforth refer to the above mentioned time windows as* **second order deception effects***. Also with respect to the empirical behaviour of an ACO algorithm we will use the notion of second order deception effects.*

The result obtained in the previous section, i.e., that the simplified model of AS applied to unconstrained problems is a local optimizer, can in general not be transfered to models of AS applied to constrained problems. The reason is that the solution construction process in ACO algorithms applied to constrained problems is a constrained sampling process (as indicated in [2]). A constrained sampling process is used, because an unconstrained sampling process (which would allow the construction of unfeasible solutions) is often not feasible in practice. As an example consider the travelling salesman problem (TSP) in undirected complete graphs with solutions represented as permutations of the n city identifiers. The number of different permutations is $n!$, whereas the unconstrained search space, which is the set of strings of length n over the alphabet $\{1, \ldots, n\}$, is of size n^n. Using Stirling's approximation (i.e., $n! \approx n^n e^{-n} \sqrt{2\pi n}$) we obtain $\frac{n^n}{n!} \approx \frac{e^n}{\sqrt{2\pi n}}$, which shows that the size of the unconstrained search space grows exponentially in comparison to the size of the constrained search space. Therefore, the probability of generating feasible solutions by an unconstrained sampling process might be extremely low.

4 Examples of Second Order Deception

Two examples of second order deception can be found in the literature. In [5], Blum and Sampels showed second order deception in the context of an ACO algorithm applied to the node-weighted k-cardinality tree (KCT) problem. However, in this case second order deception can only be detected when the AS-update rule is applied without the use of local search for improving the solutions constructed by the ants. A more serious case of second order deception is reported by Blum et. al in [3, 4] for the job shop scheduling (JSS) problem, in which is given a finite set of operations $\mathcal{O} = \{o_1, \ldots, o_n\}$ that have to be processed by machines. Each operation has a processing time assigned. The goal is to find a permutation of all operations that satisfies some precedence constraints and which is minimal in some function of the completion time of the operations. In order to apply ACO to the JSS problem, Colorni et al. [6] proposed the following CO problem model of the JSS problem: First, the set of operations $\mathcal{O}$ is augmented by two dummy operations o_0 and o_{n+1} with processing time zero. Operation o_0 serves as a source operation and o_{n+1} as a destination operation. The augmented set of operations is $\mathcal{O} = \{o_0, o_1, \ldots, o_n, o_{n+1}\}$. Then, for each operation o_i, where $i \in \{0, \ldots, n\}$, a decision variable X_i is introduced. The domain for decision variable X_0 is $D_0 = \{1, \ldots, n\}$, and for every other decision variable X_i the domain is $D_i = \{1, \ldots, n+1\} \setminus \{i\}$. The meaning of a domain value $j \in D_i$ for a decision variable X_i is that operation o_j is placed immediately after operation o_i in the permutation of all the operations to be constructed by the algorithm. We denote this CO problem model of the JSS problem by $\mathcal{P}_{\text{JSS}}^{\text{suc}}$. In order to derive the pheromone model we again introduce for each combination of a decision variable X_i and a domain value $j \in D_i$ a solution component $\mathfrak{c}_i^j$. The pheromone model then consists of a pheromone trail parameter $\mathcal{T}_i^j$ for each solution component $\mathfrak{c}_i^j$.

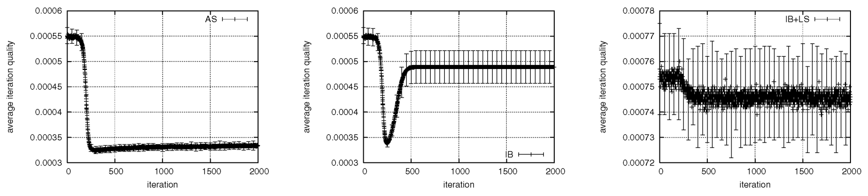
The ants' mechanism for constructing feasible solutions builds feasible permutations of all the operations from left to right. It works as follows. Let $\mathcal{I}$ denote the set of indices of the decision variables that have already assigned a value and the decision variable that receives a value in the current construction step. Furthermore, let i_c denote the index of the decision variable that receives a value in the current construction step. The solution construction starts with an empty partial solution $\mathfrak{s}^p = \langle\rangle$, with $i_c = 0$, and with $\mathcal{I} = \{0\}$. Then, at each of n construction steps $t = 1, \ldots, n$ a solution component $\mathfrak{c}_{i_c}^j \in \mathfrak{N}(\mathfrak{s}^p)$ is added to the current partial solution, where

$$\mathfrak{N}(\mathfrak{s}^p) = \left\{ \mathfrak{c}_{i_c}^j \mid o_j \in \mathcal{O}_t \right\} \quad . \tag{11}$$

In this context, $\mathcal{O}_t$ is the set of allowed operations at construction step t. This means that at each construction step we decide a domain value for the decision variable with the index i_c. When adding the solution component $\mathfrak{c}_{i_c}^j$ to $\mathfrak{s}^p$ we also set i_c to j and add j to $\mathcal{I}$. In the $(n+1)$-th construction step, the value of the last unassigned decision variable X_{i_c} is set to $n+1$. Each construction step is done according to the following probability distribution:

$$\mathbf{p}(\mathfrak{c}_{i_c}^j \mid \mathcal{T}) = \begin{cases} \frac{\tau_{i_c}^j}{\sum_{\mathfrak{c}_{i_c}^k \in \mathfrak{N}(\mathfrak{s}^p)} \tau_{i_c}^k} & \text{if } \mathfrak{c}_{i_c}^j \in \mathfrak{N}(\mathfrak{s}^p) \\ 0 & \text{otherwise .} \end{cases} \tag{12}$$

Figure 2 shows the evolution of the average iteration quality over time for several versions of Algorithm 1 based on the above described pheromone model and construction mechanism (the parameter settings of the algorithm are given in the caption of the figure). The strongest second order deception effects are obtained with the AS-update rule. However, even when using the more common IB-update rule, that is, only the best of the solutions constructed per iteration is used for updating the pheromone values, the algorithm obtains solutions at the end of a run whose average quality is lower than the average quality of the solutions in the first iteration. This shows that the harmful bias that leads to the occurrence of second order deception can be a serious problem and may cause the failure of an algorithm. The reasons for the existence of such a bias are discussed in [1].



(a) AS-update (b) IB-update (c) IB-update + LS

Fig. 2. The three graphics show the evolution of Algorithm 1 applied to the JSS problem as outlined in the text for different pheromone update rules as well as with and without the application of steepest descent local search (based on the neighborhood structure introduced by Nowicki and Smutnicki in [15]). The following settings were chosen: $n_a = 10$ (number of ants), $\rho = 0.05$ (evaporation rate), and $c = 0.5$ (initial setting of the pheromone values). The graphics show the result of 100 runs of the algorithm. The vertical bars show the standard deviation

5 Conclusions

In this paper, we first have introduced the notion of first order deception in ant colony optimization. We showed that ACO algorithms suffer from this type of deception in the same way as evolutionary algorithms do. Then, we introduced second order deception, which is a bias introduced by algorithmic components. We presented an example of second order deception from the literature. The example of the job shop scheduling problem shows that second order deception is a major issue in ACO algorithms and should be taken into account by researchers that develop ACO algorithms. This is because an ACO algorithm that suffers from strong second order deception effects might fail. Therefore, research efforts have to be undertaken towards methods for avoiding second order deception. Possibilities include choosing different solution construction mechanisms, different pheromone models, and different pheromone update rules.

Acknowledgements

This work was supported by the "Metaheuristics Network", a Research Training Network funded by the Improving Human Potential program of the CEC, grant HPRN-CT-1999-00106. The information provided 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. Marco Dorigo acknowledges support from the Belgian FNRS, of which he is a Senior Research Associate, and from the "ANTS" project, an "Action de Recherche Concertée" funded by the Scientific Research Directorate of the French Community of Belgium.

References

1. C. Blum. *Theoretical and practical aspects of ant colony optimization.* PhD thesis, IRIDIA, Université Libre de Bruxelles, Belgium, 2004.
2. C. Blum and M. Dorigo. The hyper-cube framework for ant colony optimization. *IEEE Trans. on Systems, Man, and Cybernetics – Part B*, 34(2):1161–1172, 2004.
3. C. Blum and M. Sampels. Ant Colony Optimization for FOP shop scheduling: A case study on different pheromone representations. In *Proceedings of the 2002 Congress on Evolutionary Computation (CEC'02)*, volume 2, pages 1558–1563. IEEE Computer Society Press, Los Alamitos, CA, 2002.
4. C. Blum and M. Sampels. When model bias is stronger than selection pressure. In J. J. Merelo Guervós et al., editors, *Proceedings of PPSN-VII, Seventh Int. Conference on Parallel Problem Solving from Nature*, volume 2439 of *Lecture Notes in Computer Science*, pages 893–902. Springer, Berlin, Germany, 2002.
5. C. Blum, M. Sampels, and M. Zlochin. On a particularity in model-based search. In W. B. Langdon et al., editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2002)*, pages 35–42. Morgan Kaufmann Publishers, San Francisco, CA, 2002.
6. A. Colorni, M. Dorigo, V. Maniezzo, and M. Trubian. Ant System for job-shop scheduling. *JORBEL – Belgian Journal of Operations Research, Statistics and Computer Science*, 34(1):39–53, 1994.
7. K. Deb and D. E. Goldberg. Analyzing deception in trap functions. In L. D. Whitley, editor, *Foundations of Genetic Algorithms 2*, pages 93–108. Morgan Kaufmann, San Mateo, CA, 1993.
8. M. Dorigo. *Optimization, Learning and Natural Algorithms* (in Italian). PhD thesis, Dip. di Elettronica, Politecnico di Milano, Italy, 1992.
9. M. Dorigo, V. Maniezzo, and A. Colorni. Positive feedback as a search strategy. Technical Report 91-016, Dip. di Elettronica, Politecnico di Milano, Italy, 1991.
10. M. Dorigo, V. Maniezzo, and A. Colorni. Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics – Part B*, 26(1):29–41, 1996.
11. M. Dorigo and T. Stützle. *Ant Colony Optimization.* MIT Press, Cambridge, MA, 2004.
12. D. E. Goldberg. Simple genetic algorithms and the minimal deceptive problem. In L. Davis, editor, *Genetic algorithms and simulated annealing*, pages 74–88. Pitman, London, UK, 1987.
13. D. Merkle and M. Middendorf. Modelling ACO: Composed permutation problems. In M. Dorigo, G. Di Caro, and M. Sampels, editors, *Proceedings of ANTS 2002 – From Ant Colonies to Artificial Ants: Third International Workshop on Ant Algorithms*, volume 2463 of *Lecture Notes in Computer Science*, pages 149–162. Springer Verlag, Berlin, Germany, 2002.
14. D. Merkle and M. Middendorf. Modelling the dynamics of ant colony optimization algorithms. *Evolutionary Computation*, 10(3):235–262, 2002.
15. E. Nowicki and C. Smutnicki. A fast taboo search algorithm for the job-shop problem. *Management Science*, 42(2):797–813, 1996.
16. F. Rothlauf and D. E. Goldberg. Prüfer numbers and genetic algorithms: A lesson on how the low locality of an encoding can harm the performance of GAs. In *Proceedings of PPSN-VI, Sixth International Conference on Parallel Problem Solving from Nature*, volume 1917 of *Lecture Notes in Computer Science*, pages 395–404, Springer Verlag, Berlin, Germany, 2000.