Efficient Monte Carlo methods for the simulation of catalytic surface reactions

J. J. Lukkien,* J. P. L. Segers, and P. A. J. Hilbers
Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O. Box 513, 5600 MB, Eindhoven, The Netherlands

R. J. Gelten and A. P. J. Jansen
Department of Inorganic Chemistry and Catalysis, Eindhoven University of Technology, P.O. Box 513, 5600 MB, Eindhoven, The Netherlands

(Received 28 January 1998)

Monte Carlo methods for the simulation of the dynamic behavior of surface reactions are developed, based on the chemical master equation. The methods are stated in a general framework which makes them applicable to a variety of models. Three methods are developed. A comparative analysis of the performance of the three methods, both theoretically and empirically, is included. [S1063-651X(98)08207-5]

PACS number(s): 02.70.Lq, 02.50.Ey, 02.50.Ng, 68.35.Ja

I. INTRODUCTION

With the advent of fast computers, Monte Carlo simulations have become an increasingly popular method to analyze the behavior of adsorbates on a surface. In experiments, fascinating phenomena have been observed in surface reactions, like the formation of patterns, oscillatory behavior in reaction rates, and the occurrence of reaction wave fronts on the surface. In order to analyze the connection between the microscopic reaction steps and the observed macroscopic behavior, simulations are conducted. For some 40 years, Monte Carlo methods have been successfully employed to study systems at equilibrium. Because the observed phenomena are typical for systems not at equilibrium, these classical Monte Carlo methods are not well suited. So-called *dynamical* Monte Carlo methods, in which the behavior of the system over time is simulated, are used instead. In order to perform such a simulation, a stochastic model of the chemical system is built. Elements of such a model are the crystal surface, the adsorbates, and the microscopic reaction steps that change the surface configuration over time. The behavior of this system over time is determined by the rates of these reactions. These rates are specified as probabilities and the surface configuration over time is then given by a master equation, describing the time evolution of the probability distribution of system configurations. Since the simulation follows the real time-dependent behavior, experimental results can be reproduced if the model is precise enough, in particular the relation between macroscopic and microscopic reaction rate constants can be investigated. Another interesting application is the investigation of systems at nonequilibrium phase transitions.

In the literature, numerous experiments with dynamic Monte Carlo simulations have been reported in various contexts [1–8]. The methods used are commonly presented as algorithms intertwined with the model under investigation, which makes it sometimes difficult to distinguish the two. Actually, only few methods are used. In this paper we investigate and compare three efficient methods for performing these dynamic Monte Carlo simulations. For a given model, all methods give the same results but they differ substantially in their use of computer resources, like computational power and memory. We demonstrate that the method of choice depends on the model that is being simulated and we analyze quantitatively what the difference is between the methods with respect to computational speed and memory use. We also show that it is possible to combine the methods, thus decreasing the use of resources even more.

This research has been used to build a general-purpose program for this type of simulation. The reasons for developing such a program are threefold. First, if there is an existing framework available, it is easier to try out new models and to modify existing ones. It gives a clear separation between the simulation method and the model. Second, it is a difficult and error-prone task to develop a special-purpose program for a particular model when the number of reactants and reactions is large. Third, the simulation of larger systems requires much memory and much processing capacity. This places a limit on the scale of the simulations in terms of lattice size and number of microscopic reactions. Rather than improving this for each simulated system in isolation, this is done at the higher level of the simulator. A requirement for the program follows from this: with respect to memory use and speed it must compete well with hand-coded, special-purpose programs for simulating the same model. In order to achieve this, it must support several different Monte Carlo methods.

The paper is organized as follows. In the next section we give the theoretical background for the simulations. In Sec. III three methods are developed. Two of these methods are described and used in literature; to the author's knowledge the third and most efficient one is novel. We pay particular attention to the exact time dependence of the simulations and point out how this relates to the notion of Monte Carlo steps. In Sec. IV we give an analysis of the three methods with respect to speed and memory use. Based on this we point out some improvements. A major improvement is the idea to combine two different methods in one simulation. Section V describes experiments with several models. We end with some conclusions and recommendations for further research.

*Electronic address: johanl@win.tue.nl

II. THEORETICAL BACKGROUND

We describe systems of catalytic reactions by a stochastic model. The surface is represented by a lattice: each lattice point corresponds to a surface site. In the rest of the paper we will use the term ''site'' also for lattice points. A lattice point can assume a number of distinct values that stand for the various adsorbates (an empty site is represented by a special adsorbate). The lattice together with values for all its points is called a *configuration*. A *reaction* changes a configuration into a new one. We characterize a reaction by these two configurations [9]. If in a certain configuration a reaction is possible we say that it is *enabled*. The evolution of the system over time is described by the chemical *master equation*, which is derived from first principles [10–12].

$$\frac{dP(c,t)}{dt}=\sum_{c'\neq c}\left[P(c',t)k_{c'c}-P(c,t)k_{cc'}\right]. \tag{1}$$

In this equation, $P(c,t)$ denotes the probability to find the system in configuration c at time t; $k_{cc'}$ is the transition probability of the reaction that transfers c into c' ($k_{cc'}$ is zero if there is no such reaction). One may interpret this transition probability as a microscopic analog of a rate constant and we will use the term interchangeably. Generally, this rate constant depends on the temperature, expressed through an Arrhenius expression.

$$k_{cc'}=\nu_{cc'}\exp\left(-\frac{E_{cc'}}{k_BT}\right). \tag{2}$$

Here, $E_{cc'}$ is the activation energy and $\nu_{cc'}$ the preexponential factor. In this paper we assume that the rate constants do not change over time.

In order to show the correctness of the methods that we develop, we have to show that the simulated system obeys Eq. (1). However, Eq. (1) is not of a very algorithmic nature. Therefore we choose as the basis for our simulation methods the slightly stronger formulation given by Gillespie [11,13,14] in his *hypothesis for chemical kinetics*: ''Suppose that the system is in configuration c. The probability that a particular enabled reaction $c\rightarrow c'$ occurs in an infinitesimal period δt is given by $k_{cc'}\delta t$.'' We show that a simulation that obeys this hypothesis also obeys Eq. (1). From the hypothesis we derive the distribution function that determines the time until the actual reaction takes place. Assuming that there is only a single enabled reaction $c\rightarrow c'$, let stochastic variable T denote the time that it occurs. Then,

$$P(T\geqslant t+\delta t)=P(T\geqslant t)(1-k_{cc'}\delta t).$$

Rewriting and taking the limit $\delta t\rightarrow 0$ gives

$$\frac{dP(T\geqslant t)}{dt}=-k_{cc'}P(T\geqslant t).$$

Integration yields

$$P(T\geqslant t)=\exp(-k_{cc'}t).$$

Hence, with this hypothesis the time until the next reaction has a negative exponential distribution, for each individual reaction. The following analysis is therefore strongly based on properties of this distribution. The following two are quite fundamental [15].

Property 1. The negative exponential distribution with parameter (''rate'') λ is memoryless in the following sense,

$$P(T\geqslant t+t_0|T\geqslant t_0)=\frac{P(T\geqslant t+t_0 \text{ and } T\geqslant t_0)}{P(T\geqslant t_0)}$$

$$=\frac{\exp[-\lambda(t+t_0)]}{\exp(-\lambda t_0)}=\exp(-\lambda t).$$

Property 2. Let T_i, $0\leqslant i<N$ be a sequence of random variables. T_i has a negative exponential distribution with parameter λ_i. Then

$$P(\min_i T_i\geqslant t)=\exp\left(-\sum_i \lambda_i t\right).$$

In a certain configuration c of the lattice many reactions are enabled. According to property 2, the total rate of change is given by the sum of the rate constants of all these reactions, denoted by $\Sigma_{c'\neq c}k_{cc'}$. According to property 1 it is not necessary to take the time into account when these reactions became enabled, hence we can fix our discussion to the situation at a certain time t. Let δt be such that at most one reaction occurs within δt. The probability to find the system in configuration c at time $t+\delta t$ is the sum of two terms: (1) the probability to find the system in configuration c at time t multiplied by the probability to stay in this configuration during δt and (2) the probability to find the system in some other configuration c' at time t multiplied by the probability to go from c' to c during δt,

$$P(c,t+\delta t)=P(c,t)\left(1-\delta t\sum_{c'\neq c}k_{cc'}\right)$$

$$+\sum_{c'\neq c}P(c',t)k_{c'c}\delta t.$$

By taking the limit $\delta t\rightarrow 0$ this equation reduces to (1).

The derivation of the master equation from the hypothesis is similar for time-dependent rate constants. Time-dependent rate constants are useful, for example, in the simulation of temperature programmed desorption [11] or in simulations of voltammetry experiments [16].

III. THREE METHODS

With the formulation of the preceding section the problem is essentially stated as a discrete event simulation, where the events are the occurrences of reactions. Standard methods from this research area may be applied. Therefore the first method we discuss is the regular discrete event simulation (DES) [17] algorithm. In the DES algorithm a tentative time is computed for all reactions that are possible at time 0. For reaction $c\rightarrow c'$ this time has to be drawn from $1-\exp(-k_{cc'}t)$. All reactions together with their tentative times are stored in an event list. The algorithm then proceeds by repeatedly performing the following steps: select the reaction with minimal time from the event list, advance the

system time to the time of this reaction, adjust the lattice according to the reaction, and recompute the event list. This method is also known as the *first reaction method* [11,13] (FRM).

Two aspects in this algorithm need more attention: drawing a time from an exponential distribution and recomputing the event list. Drawing a time is done using the inverse function method [17,18]: if U is a random variable distributed uniformly on (0,1) and random variable Y has distribution function F that has an inverse F^{-1}, then

$$P(F^{-1}(U)\leqslant y)=P(U\leqslant F(y))=F(y)=P(Y\leqslant y).$$

Hence Y has the same distribution function as $F^{-1}(U)$. As a result, generating a random number from $1-\exp(-k_{cc'}t)$ is equivalent to computing

$$-\frac{1}{k_{cc'}}\ln(1-u),$$

where u is a number drawn from the uniform distribution. If U is distributed uniformly, $1-U$ is uniform as well and we may replace $1-u$ by u.

With respect to the event list, it is usually not necessary to recompute it completely, since a reaction comprises only a small, local change of the lattice. Hence, if in a certain configuration c two reactions $c\rightarrow c'$ and $c\rightarrow c''$ are enabled, either one of them may still be enabled after execution of the other. According to property 1 it is not necessary to recompute the tentative times for reactions that remain enabled. We conclude that recomputing the event list actually boils down to updating it: inserting reactions that have become enabled and deleting reactions that have become disabled.

Deleting the disabled reactions requires searching the event list. In particular for larger simulation models, this is intolerable for reasons of performance. Hence, rather than searching this list, we retain the disabled reactions in the list and verify, before execution of a reaction, whether it is still enabled. To that end it is not sufficient to just check the current configuration; it may be the case that the reaction has become disabled and then enabled again. For each scheduled reaction we record the time it was scheduled and for each lattice point we record the time it was last modified. This information is enough to determine whether a reaction is still enabled. The price is an extra amount of memory, proportional to the number of lattice points and some extra processing time for the comparison. We refer to this improved method as FRMb and to the standard DES algorithm as FRMa.

FRM can be regarded as the most general method in the sense that few properties of the model are used. It can be used for time-dependent rate constants as well [11]. In the two other methods below we use properties of both the exponential distribution and the model.

Property 3. Let T_i, $0\leqslant i$ be a sequence of random variables. T_i has a negative exponential distribution with parameter λ_i. In addition, let λ be a positive number satisfying $\lambda_i\geqslant\lambda$ for all i. For each i we define a Bernoulli experiment [19] with parameter λ/λ_i. Random variable B represents the number of Bernoulli trials until the first success. If $P(B=\infty)=0$ [20] then

$$T=\sum_{i=0}^{B-1} T_i$$

has a negative exponential distribution with parameter λ. [The interpretation is that in step i a time until the next event is selected from $1-\exp(-\lambda_i t)$ which is then accepted with probability λ/λ_i.]

Proof. An intuitive argument is as follows [21]. The probability that an event occurs in the period $[t,t+\delta t)$ is given by $(\lambda/\lambda_i)\lambda_i\delta t$ for some $0\leqslant i$. Hence,

$$P(T\geqslant t+\delta t)=P(T\geqslant t)(1-\lambda\,\delta t).$$

A more precise calculation is found elsewhere [22].

Property 4. In the context of property 2, interpret T_j as the time that event j occurs. Assume that a time of the next event is selected according to the negative exponential distribution of the sum of the rates, as suggested in property 2. Next, select event j with probability $\lambda_j/\Sigma_i\lambda_i$. If event j is selected, the time of occurrence is distributed according to T_j.

Proof. Let stochastic variable F_j denote the time that event j occurs using this algorithm. We have to show that F_j has the same distribution as T_j.

$$\begin{aligned}P(F_j\geqslant t+\delta t)&=P(F_j\geqslant t)(\text{no event } j \text{ in } t+\delta t)\\&=P(F_j\geqslant t)\left(1-\frac{\lambda_j}{\sum_i \lambda_i}\sum_i \lambda_i\delta t\right)\\&=P(F_j\geqslant t)(1-\lambda_j\delta t).\end{aligned}$$

Next, we observe that, although there are many different enabled reactions in a certain configuration, there are only a few reaction *types*; the enabled reactions are instances of these reaction types at different sites. Hence, there are only a few distinct values for the $k_{cc'}$, corresponding to M different types. We denote the rate constants by k_i, $0\leqslant i<M$ and we use $n_i(c)$ to denote the number of enabled reactions of type i in configuration c. In this way we can simplify the expression for the total rate as follows. When the system is in configuration c, it changes to another configuration with rate

$$C(c)=\sum_{c'\neq c} k_{cc'}=\sum_i n_i(c)k_i.$$

According to property 2 the time increment until the next reaction has to be drawn from $1-\exp[-C(c)t]$; the corresponding reaction can then be selected by choosing type i with probability $n_i(c)k_i/C(c)$ and, if type i is selected, one of the enabled reactions of this type with probability $1/n_i(c)$. Then the times of the individual reactions are again distributed correctly according to property 4. This method is sometimes called the *variable step size method* (VSSM) and is mentioned by several authors [1,2,11,13,21]. In this formulation it requires the numbers $n_i(c)$ and their sum to be known. For the purpose of selecting one of the enabled reactions of the selected type, one has a choice between recording these reactions explicitly (VSSMa) or trying the se-

lected reaction type at random on the lattice until a matching site is found (VSSMc). The latter has the advantage that no expensive modifications (deletions) of relevant data structures are necessary.

Method VSSMa could be improved if the disabled reactions were not removed from the data structure. Suppose we do so, and consider the system in configuration c at some instant. Let l_i^0 denote the number of recorded type i reactions, some of which may be disabled, and let $D^0=\Sigma_i l_i^0 k_i$. Notice that $l_i^0 \geqslant n_i(c)$ hence $D^0 \geqslant C(c)$. If we use the above procedure for selecting a reaction with $n_i(c)$ replaced by l_i^0 and $C(c)$ replaced by D^0, we have the following probability of finding one that is enabled.

$$\sum_i \frac{l_i^0 k_i}{D^0} \frac{n_i(c)}{l_i^0} = \sum_i \frac{k_i n_i(c)}{D^0} = \frac{C(c)}{D^0}. \tag{3}$$

It is, of course, possible that a disabled reaction is selected. Then we remove this reaction, adjusting the l_i^0 into l_i^1 and repeat the procedure. If, however, an enabled reaction is selected we execute it. In this way we obtain a sequence l_i^j, $j \geqslant 0$ and corresponding sequence D^j until an enabled reaction is selected. From property 3 it follows that the time until the next reaction is distributed as $1-\exp[-C(c)t]$. If the jth of such a trial leads to an enabled reaction, a specific enabled type i reaction is selected with the following conditional probability,

$$\frac{P(\text{the } j\text{th trial is a specific type } i \text{ reaction})}{P(\text{an enabled reaction is selected in the } j\text{th trial})} = \frac{l_i^j k_i}{D^j} \frac{1}{l_i^j} \Big/ \frac{C(c)}{D^j} = \frac{k_i}{C(c)}.$$

Hence, according to property 4, the individual reactions are selected with the correct probability. As far as we know, this method is novel. We call it VSSMb.

Method VSSMc is nice because there is no need for recording the enabled reactions. However, it requires the values $n_i(c)$ to be known by the program. We can avoid this and use randomization once more to obtain them. Let K denote the sum of the rate constants of the reaction types, i.e., $K=\Sigma_i k_i$, and let N denote the total number of sites on the lattice. We assume that at any site at most one instance of each reaction type can be enabled. Then we have $NK \geqslant C(c)$. We use the same method as in VSSMb above for advancing the time. A time increment is selected from $1-\exp(-NKt)$. Then an enabled reaction is executed with probability $C(c)/NK$. This is done by selecting one of the sites, each site with probability $1/N$, and reaction type i with probability k_i/K independently. The probability that an enabled reaction is found at the selected site is then

$$\sum_i \frac{n_i(c)}{N} \frac{k_i}{K} = \frac{C(c)}{NK}. \tag{4}$$

The probability that a specific type i reaction is selected is again given by a conditional probability,

$$\frac{1}{N} \frac{k_i}{K} \Big/ \frac{C(c)}{NK} = \frac{k_i}{C(c)}.$$

This is once more in accord with properties 3 and 4. We call this the *random selection method* (RSM).

Method RSM has become quite popular in the literature although the connection with real time is usually not mentioned. Since the method decouples the notion of time completely from the simulation (the time increment is drawn from a fixed distribution) there is a tendency to measure time in Monte Carlo (MC) steps. One MC step then corresponds to one trial per lattice site on average, i.e, one MC step is N trials. This definition of MC steps also allows a comparison with other simulation techniques. From our analysis we obtain that one MC step is equivalent to a time lapse taken from the following distribution,

$$\sum_{j=0}^{N-1} T_j, \quad \text{with} \quad P(T_j \leqslant t) = 1-\exp(-NKt).$$

This value can be drawn at the end of a MC step or a value from a similar distribution can be drawn at the end of the entire simulation. If we take for T_j the expected value, i.e., $1/NK$, we obtain the following relation: One MC step corresponds to $1/K$ seconds. Because of the equivalence of the three methods, this relation can be used to estimate the number of MC steps corresponding to a given amount of simulated time, for all three methods. This is of particular importance if the exact simulation parameters (the rate constants) are not known or if the exact time dependence is of no interest. In those cases the rate constants are usually rescaled to define a probability distribution, i.e., only the values of k_i/K are given, not the k_i themselves. The relation between MC steps and simulated time becomes meaningless if time-dependent rates are used.

Without using the negative exponential distribution, method RSM can also be regarded as a time discretization of Eq. (1). The time step is then $1/NK$. Since the total rate of change is $C(c)$, the probability that an enabled reaction occurs in a time step is approximated by $C(c)/NK$.

Method RSM can be compared to a *cellular automaton* [23]. In a cellular automaton, *all* sites can make a transition in each step of the simulation. Such a step in the simulation includes conflict resolution for transitions that may disable each other. For a standard cellular automaton the notion of real time is discarded: the decision of whether to make a transition is based only on information local to that site. As a result, a slowly evolving reaction at some part of the lattice has the same probability to occur as a fast reaction at another place. In order to introduce the real-time dynamics, the decision of whether to make a transition is taken with a probability that depends on the rate constant resulting in a *nondeterministic cellular automaton*. In this case the conflict resolution must be done carefully, such that Eq. (1) is still satisfied. This nondeterministic automaton resembles method RSM best.

IV. ANALYSIS AND IMPROVEMENTS

We analyze and compare the three methods with respect to two performance measures: simulation speed and memory

use. Both measures are determined by the simulated model and the size of the lattice. In our analysis we abstract from the simulated model by choosing some characteristic parameters. We compare FRMb, VSSMb, and RSM because these are obviously the best candidates. During our analysis we describe in some more detail the basic steps of the algorithms. We use the results to point out some improvements.

A. Memory use

In all three methods, the reaction types and the current configuration of the lattice must be stored. Method RSM requires nothing else; for FRMb and VSSMb it is also necessary to record for each site the time of the last modification. Since recording such a time is more expensive than recording just the configuration, this extra overhead is significant. For example, if $N=1024^2$ and each site requires one byte to record the adsorbates, we need 1 Mbyte to store the configuration. Storing a time of last modification requires at least 4 bytes per site giving an extra 4 Mbytes.

Besides this, methods FRMb and VSSMb both rely on storing the enabled reactions in a data structure. This data structure may contain disabled reactions as well. Important questions are (1) Does the size of this data structure influence the performance of the algorithm? (2) How large can it become? Is it possible that it grows beyond limit? With respect to the second question, we note that the number of enabled reactions is limited to at most one of each type per site. If M is the number of types this limit is given by NM. Therefore we try to relate the number of stored reactions to the number of enabled reactions. If necessary, we can effectively limit the number of stored reactions by performing ''garbage collection'' at regular intervals. During a garbage collection, all disabled reactions are removed from the data structure. Later, we will find that garbage collection also increases the speed of the algorithm.

In order to answer the first question, we analyze the behavior of VSSMb in more detail. We do this by looking at averages and assuming that the simulated system is at equilibrium so that we can reason about the average number of enabled reactions [24], denoted by $e^*=\Sigma_i n_i^*$. In addition, let n_e denote the average number of reactions that become enabled and n_d the average number of reactions that become disabled when a reaction is executed. Each time a reaction is executed, n_e enabled reactions are added to e^* and n_d+1 are subtracted. Since this should not alter e^* we obtain

$$n_e=n_d+1. \tag{5}$$

The value of e^* is proportional to the size of the lattice. Furthermore it will depend strongly on the model, in particular, on the number of reaction types that can be enabled at a site at the same time. If two different models for the same system exist we prefer the one where this number is smaller,

$$e^*=\xi_0 N g(M), \tag{6}$$

where ξ_0 is a constant of proportionality and g is an increasing function. The increase of e^* as function of M corresponds to an increase of n_e with M, resulting in more work per executed reaction and, hence, in a slower simulation. Notice that properties concerning this (average) number of enabled reactions are properties of the model and do not depend on the simulation method.

Next we analyze the relation between these model parameters and the size of the data structure. Initially, the data structure is filled with the enabled reactions only. In each step of the procedure, a reaction is selected and removed from the data structure. The probability of selecting an enabled reaction is given by Eq. (3), which we call the *enabling efficiency*, ϵ. Its value depends on the contents of the data structure,

$$\epsilon=\frac{\sum_i n_i^* k_i}{\sum_i l_i k_i}. \tag{7}$$

Let h_j denote the average number of stored reactions (i.e., $\Sigma_i l_i$), after j trials and ϵ_j the corresponding efficiency. From the description we obtain the following recurrence:

$$h_0=e^*, \quad h_{j+1}=h_j+\epsilon_j n_e-1. \tag{8}$$

From Eq. (7) we observe that h_j is inversely proportional to ϵ_j. From this negative feedback we conclude that h_j tends to or oscillates around some stable value $h^*=\Sigma_i l_i^*$. From Eq. (8) we obtain that in the limit

$$\epsilon=\frac{1}{n_e}. \tag{9}$$

Hence, h^* is obtained as a solution of

$$\frac{1}{n_e}=\frac{\sum_i n_i^* k_i}{\sum_i l_i^* k_i}. \tag{10}$$

The behavior of method FRMb is more difficult to analyze but because of the algorithmic equivalence of the two methods we assume that the same results hold [i.e., Eqs. (9) and (10)]. In fact, the only difference between the two methods is that in FRMb the distribution of the reaction times is stored together with the reactions themselves.

Methods FRMb and VSSMb differ in the way data is stored. For FRMb, we need to store the following per enabled reaction: (1) (a reference to) the reaction type, (2) the site where it is enabled, (3) the time it is scheduled to be executed, and (4) the time it became enabled.

In order to allow efficient access to the reactions, they need to be stored in a *priority queue* [25]. Such a data structure allows inserting an element and selecting one with the minimum time in a time proportional to the logarithm of the size of the queue. Depending on the implementation there is some memory overhead for the implementation of the priority queue itself.

For VSSMb we do not need to store the type or the reaction time. The reactions can simply be stored in an array per type containing the site and the time it became enabled. Therefore VSSMb uses significantly less memory than FRMb.

Summarizing, we have established that RSM consumes the least amount of memory. Both FRMb and VSSMb need a potentially large but finite data structure to store the enabled reactions. Since FRMb needs to store more information per reaction it requires more memory. Finally, even if there are many disabled reactions in this data structure, the enabling efficiency is constant according to Eq. (9). Garbage collection is thus useful to limit the size of the data structure, to speed up access to the elements, and to increase the enabling efficiency temporarily.

B. Processing time

All three methods use a number of basic operations: drawing a random number, pattern matching, modification of the lattice, and adaptation of data structures containing the enabled reactions. We express the expected processing time per reaction of the three methods as functions of the times associated with these basic operations, and simplify whenever that seems appropriate.

We start again with method RSM. In each step (trial) of the method, we have the following operations.

(1) Select a site.
(2) Select a reaction type.
(3) Check if the reaction is possible at the selected site. If so, change the lattice accordingly.
(4) Advance the time.

Per trial in method RSM we need to select two random numbers, select a reaction from a collection of M, compare the pattern at the selected site, and adjust the lattice if a match is found. The probability of finding such a match is [cf. Eq. (4)] given by C^*/NK, where $C^*=\Sigma_i n_i^* k_i$, the average rate of change of the system at equilibrium. We call this probability the *matching efficiency*. This means that the average number of trials per executed reaction is given by NK/C^*. Selecting a reaction out of M possible ones can be done in a time proportional to $\ln M$, either by storing the reaction types in a binary tree or by using ''binary search'' [25,26]. For the time per simulated reaction we obtain

$$T_R=\frac{NK}{C^*}\left(2\tau_r+\tau_e+\tau_t \log_2 M+\tau_p+\frac{C^*}{NK}\tau_p\right),$$

where the τ's are the average times associated with the following computations.

τ_r: draw a random number;

τ_e: draw a number from the negative exponential distribution;

τ_t: one step in a tree walk (left or right branch);

τ_p: pattern match or modification of lattice according to pattern.

For method FRMb, we have the following operations.

(1) Select and remove the stored reaction with minimal time stamp.
(2) Verify whether the selected reaction is still enabled.
(3) If so, adjust the lattice, adjust the time, search for new enabled reactions, select a time for them and store them.

The time required to select a reaction from the priority queue is proportional to the logarithm of its size, h^*.

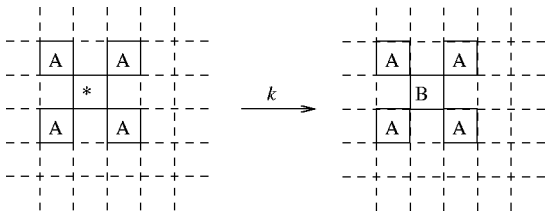

FIG. 1. An example of a reaction specification. This reaction is enabled at a certain site if the left pattern matches: an empty site (''*'') with the next-nearest-neighbor sites inhabited by species A. Then an adsorption of species B takes place with rate constant k. Notice that only the empty site changes; the A's in the specification are used to specify lateral interaction.

Both enabling checking and adjusting the lattice takes time τ_p. In a fraction $1/n_e$ [cf. Eq. (9)] of the cases, the reaction is actually possible. Then, n_e new reactions become enabled that have to be stored in the queue and for which times have to be determined. The remaining term is the computational effort in searching these new enabled reactions. In order to evaluate that we must be more precise about the representation of reaction types.

(4) A reaction type is specified by two patterns: the *source* pattern and the *target* pattern. A reaction is enabled at a certain site when this source pattern matches; if it is executed, the source pattern is replaced by the target pattern. An example is given in Fig. 1. It concerns the adsorption of a species B with rate constant k. The reaction is enabled only if an empty site, denoted by an asterisk, has four A's as next-nearest neighbors (this is an example of a reaction where lateral interaction plays a role). The A's are not modified by the reaction; they occur in the pattern only to specify the conditions under which the reaction takes place. Presumably, the complete specification would contain reactions for all possible occupations for these next-nearest-neighbor sites but with different values for k. When a reaction is executed, we have to look at the sites that were modified. As a result of this modification, some other reaction may have become enabled, viz., if the site occurs in its source pattern. Therefore we have to match each source pattern with each modified site at all possible places in this pattern. (If there are multiple modified sites we have to be careful with duplicates but we ignore that for simplicity.) The total number of pattern matches therefore amounts to the average number of modified sites per reaction (n_m) multiplied by the sum of the sizes of all source patterns (Mn_s, with n_s the average size of a source pattern). Putting this together, we obtain the following expression for the time per simulated reaction.

$$\begin{aligned}T_F&=n_e\Big(\tau_t \log_2 h^*+\tau_p\\&\quad+\frac{1}{n_e}[\tau_p+n_m M n_s \tau_p+n_e(\tau_t \log_2 h^*+\tau_e)]\Big)\\&=n_e(2\tau_t \log_2 h^*+\tau_e+\tau_p)+\tau_p(1+n_m M n_s).\end{aligned}$$

Finally, we look at VSSMb. Per trial in VSSMb the following operations are performed.

(1) Select a time till the next reaction and adjust the time.
(2) Select a reaction type.
(3) Select and remove a reaction of the selected type.
(4) Verify whether it is still enabled.
(5) If so, adjust the lattice, search for new enabled reactions, and store them.

Selecting a reaction type can be done in a time proportional to $\log_2 M$ if the reaction types are stored in a binary tree. Selecting a reaction of the selected type can be done in one step. However, assuming that the reactions are stored in a tree, removing the reaction means adjusting the tree labels that represent the rate constants of the corresponding reactions [the l_i in Eq. (3)]. This takes a time proportional to $\log_2 M$. A similar remark holds for storing a reaction. For VSSMb we therefore have the following expression:

$$T_V = n_e\left(\tau_e + 2\tau_r + 2\tau_t \log_2 M + \tau_p + \frac{1}{n_e}(\tau_p + n_m M n_s \tau_p + n_e \tau_t \log_2 M)\right) = n_e(3\tau_t \log_2 M + \tau_e + 2\tau_r + \tau_p) + \tau_p(1 + n_m M n_s).$$

When comparing the three methods we observe that the speed of method FRM depends on the lattice size through $\ln h^*$; the speed of neither RSM nor VSSMb depends on the size of the lattice. This is an important characteristic of a method. Method RSM depends strongly on the value of C^*. If this is small compared to the maximal rate of change, NK, a large number of trials per reaction will be performed. This is the case when few reactions are enabled or when the rate constant of one reaction type dominates the other ones by orders of magnitude. In general such a situation occurs for complex models with many reaction types. Then it is likely that two or more reactions proceed with a different speed, for example, in models containing both reaction and diffusion. Another source of inefficiency is the case that the reaction types exclude each other, resulting in few reactions per site that can be enabled. We expect this method to perform reasonably for particular, well-chosen models but we expect it to perform badly in general.

Methods FRMb and VSSMb have the same time component for finding the newly enabled reactions. VSSMb needs more random numbers per trial; access to the relevant data structures will in general be faster for VSSMb, since the number of enabled reactions will be much larger than the number of reaction types.

We conclude that method VSSMb is the best general method of the three when rate constants do not change over time; otherwise, FRMb should be chosen. In some specific cases it may be better to use RSM instead of VSSMb, in particular if memory is a problem or if the model is such that only few trials are performed per reaction. There is, however, much room for improvement in these two methods as we discuss below.

C. Improvements

The simulated model determines the speed of the simulation, for each of the methods. However, there may be different formulations for essentially the same model and then the formulas give some hints which formulation to choose: it is important to have n_e and M as small as possible, and to work with small neighborhoods.

For VSSMb and FRMb a substantial amount of the computation time goes into pattern matching. Much of this pattern matching is superfluous since, in general, not every reaction type can enable all the other ones, and certainly not at all modified points. For example, the reaction in Fig. 1 will not enable reactions that do not have a B in their source patterns. The time spent in searching for enabled reactions can then be reduced by computing this enabling relation between reaction types before the actual simulation starts. Additional improvements through preprocessing avoid comparisons of which the outcome is already known. This reduces the computation time substantially as we will show later.

An improvement for RSM would be the reduction of the total rate, NK. Suppose that there are two reaction types, i and j with the same rate constant, $k_i = k_j$, and suppose that it is impossible that both of them are enabled at the same site. Hence, $n_i(c) + n_j(c) \leq N$, for all configurations c. This implies

$$C(c) \leq N(K - k_i) = NK'.$$

We make a new reaction type: ''i or j'' with rate constant k_i and remove both types i and j. The probability to find an enabled reaction is now as follows.

$$\sum_{l, l\neq i, l\neq j} \frac{n_l(c)}{N}\frac{k_l}{K'} + \frac{n_i(c) + n_j(c)}{N}\frac{k_i}{K'} = \frac{C(c)}{NK'}$$

and the probability to select a specific type l reaction

$$\frac{1}{N}\frac{k_l}{K'} \Big/ \frac{C(c)}{NK'} = \frac{k_l}{C(c)}.$$

Hence, we may combine reaction types that exclude each other, decreasing the total rate thus speeding up the simulation. It must be noted, however, that this increases the value of τ_p. In case the rate constants of two mutually exclusive reactions are not equal we may combine them partially, using the fact that if we include a reaction type twice with rate constants k_1 and k_2 this is identical to including this type only once with rate constant $k_1 + k_2$.

Finally, a considerable reduction of computation time may be obtained by using more than one method in the same simulation. Two (or more) methods may be combined by separating the reaction types into two groups, each group being simulated with a separate method. For each of the two groups the time of the next event or trial is determined. The group with the smallest time is selected and an event or trial is executed. If an event occurs, new times are determined for *both* groups.

An example of when this is useful is the following situation. In a system simulated with VSSMb we have two reaction types, x and y, with rate constants k_x and k_y. Type x may enable type y, k_x is much larger than k_y, and x appears to occur frequently. As a result, y is matched as frequently as x occurs. However, if y is simulated using RSM it is matched only Nk_y times per simulated second, on average. If

the latter is smaller there is too much pattern matching and, probably, much disabling and superfluous storage in the simulation using VSSMb only. Hence, we save both memory and simulation time by using RSM for y and VSSMb for x. An example of this is the simulation of a model with rapid diffusion.

V. PERFORMANCE ANALYSIS FOR SOME EXAMPLE MODELS

We present three example models, two of which we have taken from literature. We show how the models can be expressed in our terminology of reaction types and corresponding rate constants. Then we give performance figures obtained from simulations with these models. Our prime interest is in validating the theoretical results derived in the preceding section. The models serve as examples and for detailed discussions on the models themselves we refer to the relevant papers. The results we obtained from the simulation were the same as the results described in the papers. The three examples are chosen such that each highlights a part of the above discussion.

The first two models were simulated using a Toshiba Satellite Pro 430 CDT with a Pentium 120 MHz processor and 32 Mbytes of internal memory. The third model was simulated using a SUN Ultra2 with 256 Mbytes of internal memory. The simulations were done using the developed program named CARLOS. As for the current study we are not interested in the models themselves but rather in the general behavior of the methods used, we did not do extensive statistics. We present results from a few runs for lattice dimensions that are powers of 2.

In order to allow a comparison between the methods, we simulated exact time dependence, in all cases. We can improve the speed of method RSM significantly by simulating MC steps when exact time is of no interest. This improves the performance up to 25%, in particular if the matching efficiency is low.

Some general results obtained from the simulations are the following. The assumption that the behavior of FRMb and VSSMb with respect to memory use and efficiency is the same was found to be correct. In addition, the efficiencies (matching efficiency and enabling efficiency) were found to be independent of the lattice dimension, as expected.

A. CO oxidation on a single crystal surface

The model

Ziff, Gulari, and Barshad present the following model [6] for the oxidation of CO on a single crystal surface,

$$\mathrm{CO}+*\rightarrow \mathrm{CO(ads)}, \tag{11}$$

$$\mathrm{O_2}+**\rightarrow 2\mathrm{O(ads)}, \tag{12}$$

$$\mathrm{CO(ads)}+\mathrm{O(ads)}\rightarrow \mathrm{CO_2}+**. \tag{13}$$

Here, the asterisk denotes an empty site on the crystal surface, represented by a square lattice. CO and O_2 adsorb irreversibly on the surface, the adsorption of O_2 being dissociative. A neighboring CO-O pair reacts and desorbs immediately. The reaction is controlled by a single parameter y_{CO}, representing the CO concentration in the gas phase. The following Monte Carlo algorithm is given.

(1) Select a molecule: CO with probability y_{CO} and O_2 with probability $1-y_{\mathrm{CO}}$.
(2) If the molecule is CO:
  (a) Select a site.
  (b) If the site is unoccupied,
    (i) CO adsorbs,
    (ii) the four neighbors are checked in random order and if a CO-O pair is found they react and desorb.
(3) Else (the molecule is O_2):
  (a) Select a pair of adjacent sites.
  (b) If both sites are unoccupied,
    (i) O_2 dissociates and adsorbs,
    (ii) the six neighbors are checked in random order and if a CO-O pair is found they react and desorb.

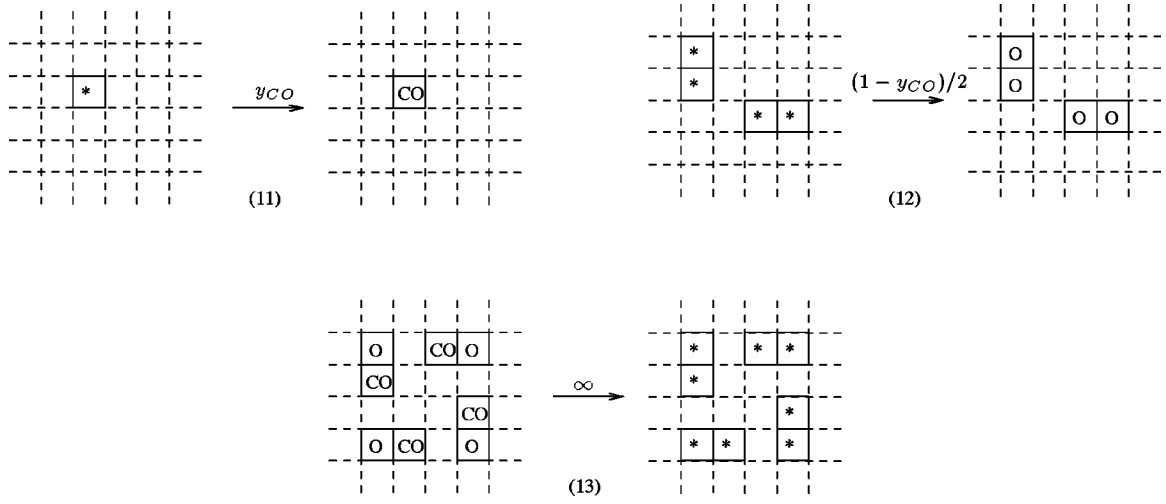

FIG. 2. Patterns (reaction types) corresponding to Eqs. (11) (one pattern), (12) (two patterns), and (13) (four patterns). The probability is distributed evenly over the possibilities.

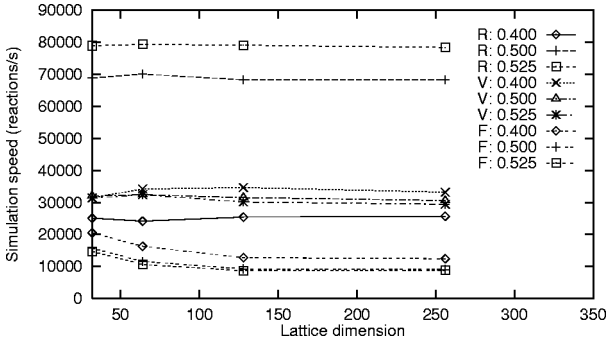

FIG. 3. Simulation speeds for the three methods, for three different values of y_{CO} and varying lattice sizes. The labels of the graphs indicate the method. FRMb performs worst, showing a degradation as the lattice size increases. RSM and VSSMb speeds do not depend on the lattice size. For lower values of y_{CO}, VSSMb performs better than RSM. All results are obtained without using garbage collection.

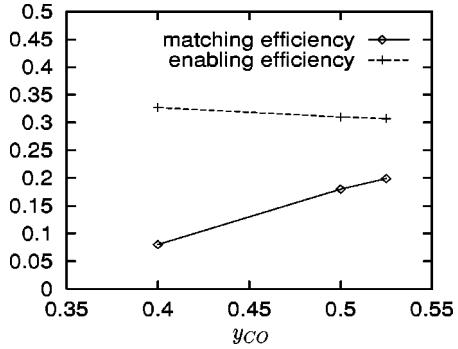

FIG. 4. The enabling efficiency is determined experimentally as the fraction of the stored enabled reactions that are actually executed [cf. Eq. (9)]. This value must be (and is found) the same for methods VSSMb and FRMb. The matching efficiency is the fraction of trials on the lattice that lead to a reaction. This determines the performance of method RSM.

The CO_2 production rate (defined as the number of CO_2 production steps per site per MC step) as a function of y_{CO} exhibits two phase transitions: a first order phase transition at $y_{CO}=0.525\pm0.001$ and a second order transition at $y_{CO}=0.389\pm0.005$. The surface becomes poisoned with either CO or O, respectively, when y_{CO} falls outside this region.

This algorithm can be rephrased as follows. Equations (11)–(13) give rise to seven reaction types in total. For Eq. (12) we have two distinct reaction types, corresponding to the situation that the pair of vacant sites is horizontally or vertically arranged. For Eq. (13) we have four such situations. This is captured in Fig. 2.

Simulation results

The program CARLOS [27] takes a model as input. The model is denoted in a similar way as the above list of reaction types. In Fig. 3 we have plotted the speed of the simulation as a function of the lattice dimension and of y_{CO}, the concentration of CO in the gas phase; the graphs are marked with the first letter of the method. We expected this speed to be independent of the size of the lattice for methods VSSMb and RSM; for FRMb it should decrease with the lattice size. This is confirmed by Fig. 3. The most important aspect of the dependency on the lattice size is that already for small lattices, $\log_2 h^*$ is an order of magnitude larger than $\log_2 M$ hence VSSMb is to be preferred. Therefore we do not discuss FRMb anymore. We observe also that there is a crossover point for preferring method VSSM above RSM. This is because variations in y_{CO} have a dramatic impact on the performance of method RSM. As we have derived, this dependence is through the enabling efficiency C^*/NK. When y_{CO} decreases, this value drops steeply (Fig. 4) which confirms the weakness of this method. For the other two methods we have a less profound fluctuation. For small values of y_{CO} the activity on the grid degrades. This results in a somewhat smaller value for n_e, giving a higher simulation speed.

Next we look at memory use. The results in Fig. 3 are without using garbage collection. For the 128^2 lattice we used some 40 kilobytes for method RSM and 2 Mbytes for VSSMb and 4 Mbytes for FRMb. By restraining the available memory of the VSSMb simulation we enforced garbage collection. Results are summarized in Table I. As expected, using garbage collection also contributes significantly to the speed of the simulation because deleting reactions through garbage collection is more efficient than through the VSSMb algorithm. In Fig. 5 we have shown the number of stored reactions during a part of these two simulations. This number tends towards an equilibrium value as predicted; the peaks correspond to garbage collection. In the right picture we have shown the enabling efficiency. A garbage collection increases this efficiency temporarily; frequent garbage collection increases the average value.

B. Oscillations in $A+B\rightarrow AB$

Fichthorn, Gulari, and Ziff [7,8] investigate the following model for simulating oscillatory behavior in reaction rates.

$$A+*\rightleftharpoons A(\text{ads}), \tag{14}$$

$$B+*\rightleftharpoons B(\text{ads}), \tag{15}$$

$$A(\text{ads})+B(\text{ads})\rightarrow AB+**. \tag{16}$$

The behavior of the system is assumed to be reaction limited. The rate of adsorption is taken to be infinite. The single parameter controlling the system is the rate constant of desorption, p_d. We refrain from repeating the used algorithm but only give the reaction types and rate constants in Fig. 6. For relatively large values of p_d a steady AB production rate is observed. When the value of p_d is decreased, this rate starts to oscillate; when p_d approaches zero, the system

TABLE I. The effect of garbage collection is that disabled reactions are discarded more efficiently than through the regular algorithm. This gives a trade-off: garbage collection (GC) must be frequent enough to ensure an increase in the enabling efficiency (see Fig. 5) but if performed too often it slows down the simulation. A gain of more than 25% is found here.

	No GC	A few times GC	Frequent GC
Memory use	2000 kbytes	850 kbytes	400 kbytes
Reactions/sec	30158	34876	38254

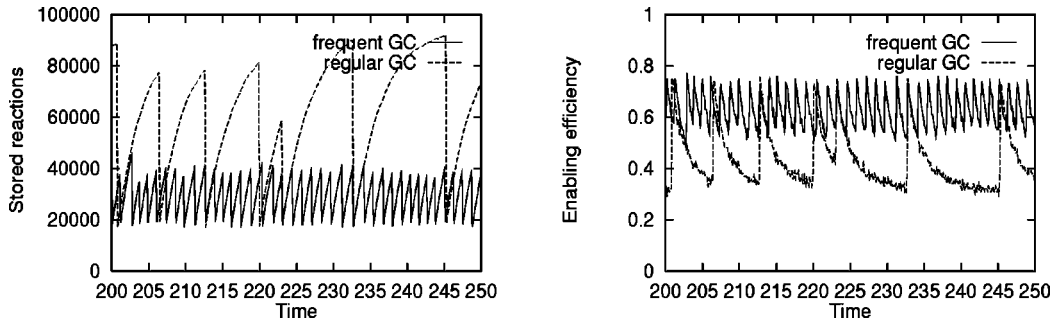

FIG. 5. The left graph gives the number of stored reactions during a part of the simulation on a 128×128 lattice. This number tends to some fixed value. When frequent garbage collection is used, this value is never reached, effectively increasing the enabling efficiency (right graph).

switches irregularly between two configurations: entirely covered with A or entirely covered with B. For $p_d=0$, poisoning with one of the adsorbates occurs.

We include this example in order to analyze the effect of the formulation of the model on the performance of the algorithms. To that end we give a different, but equivalent model. The interesting results obtained from the original model are the AB production rate and the surface coverage. For this purpose the notion of an empty site is actually superfluous. Instead of using the infinite-rate reactions we may collapse them into the patterns of Fig. 7. This reformulation increases the number of reaction types, which is harmless for method RSM but bad for the other methods. For method RSM we should also combine reactions that exclude each other. We may, for example, combine the $A\rightarrow B$ with the $B\rightarrow A$ reaction as has been done in the algorithmic description [8]. Also the possible AB production steps can be taken into four groups.

Experiments for a 64×64 grid with $p_d=0.02$ are given in Table II. Clearly, when $p_d=0.02$ RSM is superior for this model. Notice that in the new model the efficiency for RSM is lower. However, because fewer steps are required in an AB reaction, the simulation runs faster. The memory requirements for VSSMb were kept minimal through frequent garbage collection. If this is not done, the low efficiency and large number of enabled reactions per site lead to a large memory requirement. The dependence of VSSMb for the way the model is formulated is impressive. It suggests a method for developing models that allow an efficient simulation with VSSMb: if many reactions per site can be enabled, develop an equivalent model through the introduction of intermediate states (like the empty site above) and dummy reactions that execute infinitely fast.

C. Oscillations and pattern formation during CO oxidation on Pt(100)

One of the reasons to investigate the more complicated methods VSSM and FRM was that we wanted to simulate models of much larger complexity than the ones given above. Within our project, a model for the oscillation of CO oxidation on a Pt surface has been developed; it is described elsewhere [28]. It is not our intention to discuss the model or the results in detail. We only want to introduce it as an example of a more complicated model.

Oscillations in the CO oxidation on a Pt surface have been attributed to numerous mechanisms, one of them being the $1\times1\rightleftharpoons$hexagonal reconstruction of Pt(100) [29,30]. Based on this mechanism, the following model is built. Initially, the Pt surface is empty and reconstructs to the hexagonal phase. CO adsorbs on the surface and reconstruction to the 1×1 phase occurs locally if the local concentration of CO is high enough. O_2 dissociates and adsorbs only on the 1×1 surface; then it rapidly reacts with CO after which CO_2 desorbs from the surface. Reconstruction to the hexagonal phase occurs for 1×1 sites with a certain rate. The start of an oscillation is induced by desorption of two neighboring CO atoms from the 1×1 surface; this allows the adsorption of O_2.

Initially, we investigated a model without diffusion. Although a global synchronization mechanism was not part of the model, we found oscillations and pattern formation with parameter values chosen close to experimental values and in a temperature range close to the range found in laboratory observations [28].

For our current discussion, we want to focus on the influence of the choice of the method and of computing the dependency relation on the performance of the algorithm. We

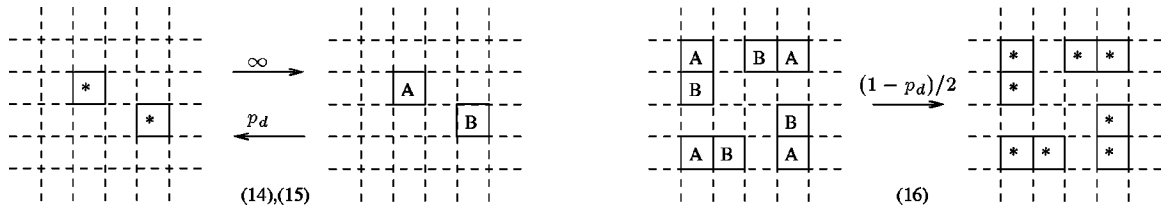

FIG. 6. Patterns (reaction types) corresponding to Eqs. (14) (two patterns), (15) (two patterns), and (16) (four patterns). A pattern on the one side is replaced by the corresponding pattern on the other side if the reaction is executed. The probability is distributed evenly over the possibilities.

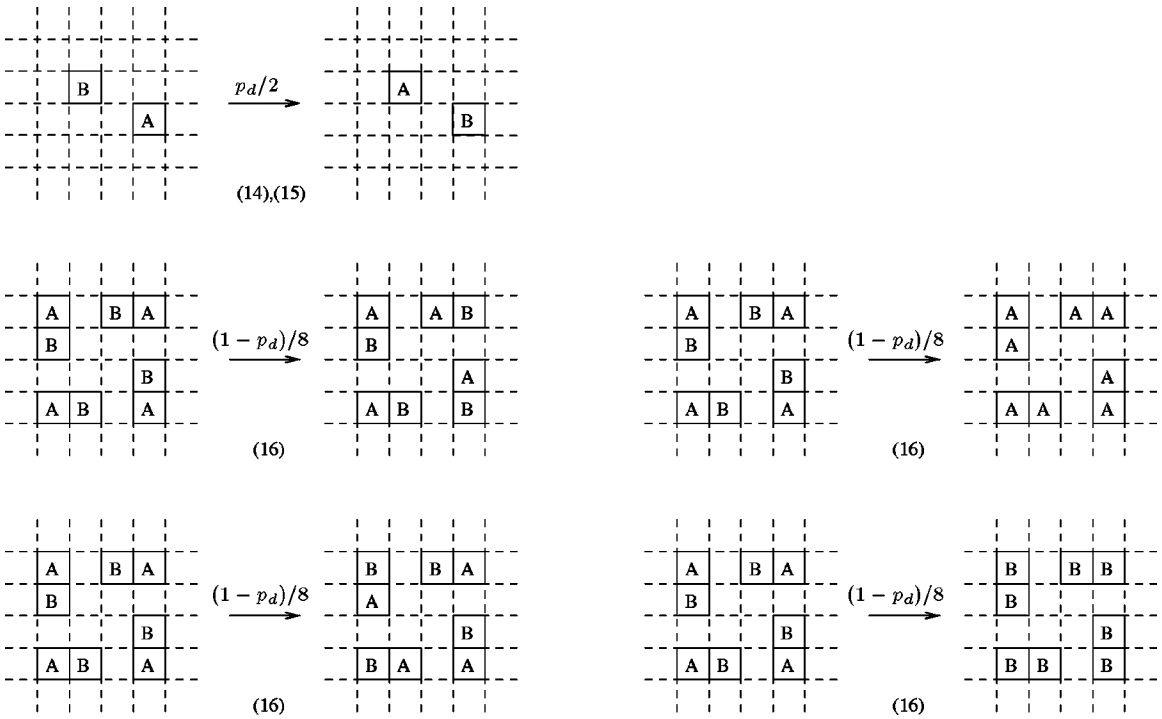

FIG. 7. Alternative formulation of the model of Fig. 6. Formation of an empty site and adsorption of a particle is taken together as a single step. Because of the possible outcomes, this increases the number of patterns significantly.

compared four different simulation methods: RSM only, VSSMb only, VSSMb after precomputing the dependencies between reaction types, and combine VSSMb and RSM using the latter for selected reaction types. The simulations were conducted on a 256×256 lattice using a temperature of 490 K. The model consists of 31 patterns (reaction types). The results are given in Table III. The column ''memory'' refers to the amount of memory required to store the reactions in VSSMb so we ignore here the amount that is needed to store the dependency relation and the extra amount per lattice point that is required to use VSSMb. Apparently, for this model method RSM is no good at all. An important reason for this is that at the given temperature the rate constants of the reactions are not in the same range. In particular, CO_2 desorption is significantly faster than the other ones but since it does not occur very frequently most time is spent in useless pattern matching for this reaction. VSSMb has a rather good performance, in particular when preprocessing is used to determine the dependencies between reactions. Since this model is much larger than the ones studied above, this analysis contributes significantly in the performance. Finally, by studying the results of the simulations we can find out which patterns are matched more often in VSSMb than they would have been using RSM for them only. This gives an indication to use RSM for such a reaction. For this model, the gain in speed was marginal; however, the memory requirement decreased.

In a second model we included diffusion of CO over the lattice. Since diffusion is much more rapid than the other reactions it becomes the dominant part in the simulation. The

TABLE II. The effect of the formulation of the model. Two formulations for the $A+B\rightarrow AB$ model differ substantially in performance. Method VSSM is sensitive for the number of enabled reactions per site; this determines the number of enabled reactions per executed reaction. Method RSM is sensitive for the matching efficiency. Although this number is lower in the new model, more AB productions can be simulated per second since fewer microscopic steps are needed.

	Original model		New model	
	RSM	VSSM	RSM	VSSM
AB production/sec	32729	14985	52940	6765
Efficiency	0.227	0.318	0.190	0.063
Memory use	4 kbytes	200 kbytes	4 kbytes	300 kbytes

TABLE III. Comparing the performance of several methods for the model describing oscillations and pattern formation during CO oxidation on Pt(100). Without diffusion in the model RSM performs badly, mainly because CO_2 production dominates in speed but does not occur very frequently. In VSSMb we see clearly the influence of precomputing the dependency relations between reactions (the rows marked w.p. in the table). When diffusion is present, RSM performs a lot better: diffusion is the fastest reaction and occurs often. Because of this rapid diffusion, disabling of reactions also occurs frequently, which is why combining the two methods pays.

	No diffusion		With diffusion	
	Reactions/sec	Memory	Reactions/sec	Memory
RSM	107	0	5464	0
VSSMb	6078	1 Mbyte		
VSSMb w.p.	30 409	1 Mbyte	17678	3 Mbytes
VSSMb w.p. +RSM	34 844	160 kbytes	29555	1.5 Mbytes

model grew to 51 reaction types. Now method RSM appears to perform much better, the reason being that the most selected reaction is diffusion and that diffusion is often enabled. We conducted three simulations (see Table III again): RSM only, VSSMb only (with preprocessing), and a hybrid simulation. Now it appears that switching to RSM for some reaction types improves the speed significantly. Notice also that, since all methods give a reasonable speed, this allows a trade-off between memory use and speed.

VI. CONCLUSIONS

We have presented three Monte Carlo algorithms for the dynamic, time-dependent simulation of chemical processes. We have restricted ourselves to time-independent reaction rate constants; a generalization to time-dependent rates is possible [11].

The algorithms were clearly separated from the simulated models. We think that this separation is important. In the literature many methods for Monte Carlo simulations are presented without a clear reference to a common basis. For some of these methods we have shown how they can be cast in the general framework of a stochastic process, presented in Sec. II. The separation has a second advantage. It allows the development of a general-purpose program and, after that, allows one to focus more on developing the models rather than a combination of a model and a method. This supports the construction of more complicated simulation models as well.

We have compared the three methods with respect to memory use and speed. For simple models with time-independent rate constants, not too much variation in reaction speeds, and with a high activity, method RSM gives good results. Further advantages are that it needs little memory and admits a simple implementation. Method VSSMb is a good alternative though its implementation is more elaborate and it needs more memory. For larger models, VSSMb is the better choice. If the variety in reaction speeds is large, a significant improvement is obtained by combining VSSMb and RSM into one simulation. In fact, our analysis demonstrated that each model requires its own method. The third method, FRMb, though slower is the most general because it allows the simulation of systems with time-dependent rate constants. In this respect, each of the three methods has a particular contribution. We have shown the impact on the performance of garbage collection and of the way the model is formulated. For larger models a dependency analysis of the reactions also yields a significant improvement.

The developed methods are applicable to a wider class of systems than just chemical surface reactions. They are particularly efficient for surface reactions because of the fact that events can be classified into types.

The developed program is a general tool for this type of Monte Carlo simulation. The input is a MC model as described in Sec. III, together with the method to be used, which is quite flexible. However, the method of using pattern matching for determining the enabled reactions is not always suitable. Consider, for example, the problem of simulating diffusion of the adsorbate where the diffusion rate of a particle depends on the occupation of its neighbor and nearest-neighbor sites. This results in a large set of reaction types, corresponding to all possible patterns. An alternative is to treat diffusion as a special case. This is done by several authors [31,32].

For large lattices the simulation time and the memory requirement grow to be very large. An implementation on a parallel platform would allow us to perform simulations for these larger systems. The difficulty then is that the description of the stochastic process is intrinsically sequential in nature and this is reflected in the methods. Only method RSM possibly allows some parallelism, since the choice of reaction and site does not depend on the current configuration. A parallel algorithm for the model of Ziff *et al.* [6] has been developed [33], based on this observation. Some other work has been done exploiting this potential, based on an assumption of homogeneity [34]. Parallel simulation techniques like TIME WARP [35] seem to be less applicable for this type of simulation because of the high disabling rate. Currently we are investigating a parallel implementation of VSSMb based on a similar technique as described in the work mentioned above.

Note added in proof. Recently, we learned that selecting one reaction type out of M possible ones can even be done in constant time using the method of ''aliasing,'' described in Ref. [26]. In our analysis of RSM and our experiments we did not use this and, since M is rather small in the models we studied, we do not expect much improvement for those models. For VSSMb we cannot use this method of aliasing since in VSSMb the weights used in the selection are not constant during the source of the simulation. The algorithm suggested in Ref. [26] to deal with varying rates will be an improvement only for very large values of M.

[1] A. B. Bortz, M. H. Kalos, and J. L. Lebowitz, J. Comput. Phys. **17**, 10 (1975).
[2] A. M. Bowler and E. S. Hood, J. Chem. Phys. **94**, 5162 (1981).
[3] H. C. Kang and W. H. Weinberg, Phys. Rev. B **38**, 11543 (1988).
[4] H. C. Kang and W. H. Weinberg, J. Chem. Phys. **90**, 2824 (1989).
[5] P. A. Maksym, Semicond. Sci. Technol. **3**, 594 (1988).
[6] R. M. Ziff, E. Gulari, and Y. Barshad, Phys. Rev. Lett. **56**, 2553 (1986).
[7] K. A. Fichthorn, E. Gulari, and R. M. Ziff, Phys. Rev. Lett. **63**, 1527 (1989).
[8] K. A. Fichthorn, E. Gulari, and R. M. Ziff, Chem. Eng. Sci. **44**, 1403 (1989).
[9] Usually, a reaction changes only a small neighborhood on the lattice. In a Monte Carlo method this may be used to obtain a significantly faster algorithm. We come back to this later.
[10] N. G. van Kampen, *Stochastic Processes in Physics and Chemistry* (North-Holland, Amsterdam, 1992).
[11] A. P. J. Jansen, Comput. Phys. Commun. **86**, 1 (1995).

[12] A. P. J. Jansen, Appl. Catal. A. **160**, 99 (1997).
[13] D. T. Gillespie, J. Comput. Phys. **22**, 403 (1976).
[14] D. T. Gillespie, J. Phys. Chem. **81**, 2340 (1977).
[15] W. Feller, *An Introduction to Probability Theory and Its Applications* (Wiley, London, 1970).
[16] M. T. M. Koper, J. J. Lukkien, A. P. J. Jansen, R. A. van Santen, and P. A. J. Hilbers, J. Chem. Phys. (to be published).
[17] I. Mitrani, *Simulation Techniques for Discrete Event Systems* (Cambridge University Press, Cambridge, England, 1982).
[18] D. E. Knuth, *Seminumerical Algorithms*, 2nd ed. (Addison-Wesley, Reading, MA, 1981).
[19] A Bernoulli experiment with parameter p is an experiment that has two possible outcomes, usually called s(uccess) and f(ailure). The parameter gives the probability that the outcome is s.
[20] The precise formulation is
$$\lim_{k\to\infty} \prod_{i=0}^{k} (1-\lambda/\lambda_i)=0.$$
[21] K. Binder, in *Monte Carlo Methods in Statistical Physics*, edited by K. Binder, Topics in Current Physics Vol. 7 (Springer, Berlin, 1979).
[22] J. P. L. Segers, Ph.D. dissertation, Eindhoven University of Technology, 1998.
[23] J. P. Boon, D. Dab, R. Kapral, and A. Lawniczak, Phys. Rep. **273**, 55 (1996).
[24] This is indeed a simplifying assumption since this type of simulation is used in particular for systems not at equilibrium. We assume, however, that variations in the time evolution of the simulated system (like the occurrence of oscillations) occur at a much slower scale than the adaptation of the data structures to these phenomena. Then this analysis is applicable to a small (simulated) period.
[25] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms* (Addison-Wesley, Reading, MA, 1983).
[26] D. E. Knuth, *Sorting and Searching* (Addison-Wesley, Reading, MA, 1997).
[27] Some information about the program can be obtained via http://www.win.tue.nl/cs/pa/johanl/carlos. We just started to create this web site.
[28] R. J. Gelten, A. P. J. Jansen, R. A. van Santen, J. J. Lukkien, J. P. L. Segers, and P. A. J. Hilbers, J. Chem. Phys. **108**, 5921 (1998).
[29] R. Imbihl, M. P. Cox, G. Ertl, H. Muller, and W. Brenig, J. Chem. Phys. **83**, 1578 (1985).
[30] M. P. Cox, G. Ertl, R. Imbihl, and J. Rustig, Surf. Sci. **134**, L517 (1983).
[31] R. Danielak *et al.*, Physica A **229**, 428 (1996).
[32] M. Tammaro, M. Sabella, and J. W. Evans, J. Chem. Phys. **103**, 10277 (1995).
[33] J. Segers, J. J. Lukkien, and P. A. J. Hilbers, in *High-Performance Computing and Networking, Proceedings HPCN Europe, 1996*, edited by H. Liddell, A. Colbrook, B. Hertzberger, and P. Sloot, Lecture Notes in Computer Science Vol. 1067 (Springer, New York, 1996), p. 243.
[34] N. Haider, A. K. Souheil, M. R. Wilby, and D. D. Vvedensky, Comput. Phys. **9**, 85 (1995).
[35] *Advances in Parallel and Distributed Simulation*, edited by V. Madisetti, D. Nicol, and R. Fujimoto, Simulation Series Vol. 23 (Society for Computer Simulation, 1990).