Stochastic Generator of Chemical Structure. 3. Reaction Network Generation

Jean-Loup Faulon*

Computational Biology and Materials Technology Department, Sandia National Laboratories, Albuquerque, New Mexico 87185-1111

Allen G. Sault

Catalytic and Porous Materials Department, Sandia National Laboratories, Albuquerque, New Mexico 87185-1349

Received March 25, 2000

A new method to generate chemical reaction network is proposed. The particularity of the method is that network generation and mechanism reduction are performed simultaneously using sampling techniques. Our method is tested for hydrocarbon thermal cracking. Results and theoretical arguments demonstrate that our method scales in polynomial time while other deterministic network generators scale in exponential time. This finding offers the possibility of investigating complex reacting systems such as those studied in petroleum refining and combustion.

I. INTRODUCTION

Motivated by applications in synthesis design, combustion, and petrochemical refining, reaction network generation has been a prolific field of research for the past 25 years. As reviewed in Ugi et al. paper,[1] reaction network generation is based on three types of techniques. Empirical techniques, which strategies are based on data from reaction libraries, semiformal methods based on heuristic algorithms, where reactions are derived from a few mechanistic elementary reactions, and formal techniques, based on graph theory. Ugi et al. argue that formal techniques are general enough to be applicable to any type of reacting system in organic or inorganic chemistry, furthermore formal techniques are the only methods capable of generating new reaction mechanisms and therefore elucidating unresolved chemical processes.

While formal techniques are robust their computational scaling limits their applicability to real reacting systems. Indeed as it has been shown by several authors,[1−9] for many processes, the number of reactions and intermediate species that are created generally scale exponentially with the number of atoms of the reactants. Two views of approaching this problem have been proposed. One is to wait until sufficient computational power becomes available. However, the exponential scaling of formal techniques makes it improbable that we will ever reach enough computing power to solve the problem. The other view, which is considered in the present paper, is to reduce the reaction mechanism. The question raised when reducing a mechanism is how to choose a reaction subset such that it describes correctly the dynamic behavior of the studied reacting system. Reduction strategies in the area of combustion modeling have been reviewed by Frenklach,[5] and these are quite general and applicable to other systems. According to Frenklach there are five types of reduction strategies: (1) global reduction, (2) response modeling, (3) chemical lumping, (4) statistical lumping, and (5) detailed reduction. Global modeling techniques transform a complete reaction scheme into a small number of global reaction steps. Global techniques comprise ad-hoc methods such as empirical fitting, reduction by approximations, and lumping. All global techniques are specific to a particular problem and cannot be generalized. Response modeling techniques consist of mapping model responses and model variables through functional relationships. Generally, models responses are species concentrations and model variables are the initial boundary conditions of the reacting mixture and the model parameters, such as rate coefficients and transport properties. The solution mapping method provides a general procedure to solve model response. In this method, model responses are expressed as simple algebraic functions (usually polynomials) in terms of model variables. These algebraic functions are obtained by using either computer experiments or experimental data. As with global techniques, response modeling solutions are problem specific since they require data to build algebraic functions. Chemical lumping and statistical lumping were both developed for polymerization-type reactions. Chemical lumping models are used when a polymer grows by reaction between the polymer and monomer species. The lumping strategy is guided by similarity in chemical structure or chemical reactivity of the reacting species. The main assumption of chemical lumping is that the reactions describing the polymer growth are essentially the same and the associated thermochemical and rate parameters exhibit only a weak dependence on the polymer size. Statistical lumping is used when polymer−polymer interactions are of concern, such as in soot formation, silica powder growth, and metal oxide growth. Such processes are described by the Smoluchowski equation. The integration of the Smoluchowki equation encounters severe prohibitive demands on computer time and memory and statistical approximation of the equation have been proposed. Both chemical and statistical lumping methods are specific

* Corresponding author e-mail: jfaulon@cs.sandia.gov.

10.1021/ci000029m CCC: $20.00
Published on Web 05/08/2001

to polymerization reactions. Finally, the detailed reduction technique consists of identifying and removing noncontributing reactions. An effective reduction strategy is to compare the individual reaction rates with the rate of a chosen reference reaction. The reference reaction being for instance the rate-limiting step or the fastest reaction. The detailed reaction reduction approach is a general technique and is a priori applicable to any reacting system.

The goal of this paper is to propose an original technique for reaction network generation that is computationally tractable and general enough to be applicable to real processes, such as combustion, petroleum refining, and retrosynthesis. To achieve such a goal, formal generation techniques must be used since with the above chemical processes not all reactions are known and there is a need to create new reaction mechanisms. However, as mentioned earlier formal techniques scale exponentially with the problem size and therefore reduction methods need to be applied. As discussed above, the only reduction technique applicable to general systems is the detailed reduction method. The caveat of the method, however, is that the entire network must be known prior to reduction. Therefore, network generation and reduction cannot be processed in sequence if our goal is to derive a computationally tractable technique.

In section II, we outline a new method where network generation and reduction are performed simultaneously. More precisely we give four algorithms: deterministic-network-generator (*DNG*), random-sampling-network-generator (*RSNG*), concentration-sampling-network-generator (*CSNG*), and MC-sampling-network-generator (*MCNG*). While the first algorithm is deterministic and similar to techniques previously reported, the three sampling algorithms are stochastic in nature and appear to be the first efficient (i.e., polynomial-time) methods ever published for reaction network generation. Susnow et al.[8] propose a technique similar to *CSNG* using rates instead of concentrations for mechanism reduction. However, the computational scaling of Susnow et al. technique has not been reported, furthermore, it is improbable that their algorithm scales in polynomial time since their technique does not explicitely limit the number of species and reactions allowed in the network. Our *CSNG*, and *MCNG*, algorithms require on-the-fly computation of the rate constants of the reactions generated. In section III, we present a quantitative structure−property relationship (QSPR) approach to compute rate constants at low computational cost. In section IV, we theoretically prove that our sampling generation-reduction algorithms no longer scale exponentially with the problem size but scale polynomially. Finally, in section V, in the context hydrocarbon thermal cracking, we test our sampling algorithms versus experimental results and deterministic algorithms. We also compare the running-time of our algorithms with the theoretical computational complexity calculations of section IV.

II. METHOD

The proposed network generator is based on the Dugundji-Ugi theory.[10] According to this theory compounds and reaction are represented by bond electron (*be*-) and reaction (*r*-)matrices. In a given *be*-matrix representing a compound the *i*th row (and column) is assigned the *i*th atom of the compound. The entry b_{ij} (b_{ji}, $i \neq j$) of the *i*th row and *j*th column of a *be*-matrix is the bond order of the covalent bond between atoms *i* and *j*. The diagonal entry b_{ij} is the number of free valence electron for atom *i*. The reader may noticed that the adjacency and connectivity matrices used in chemical information differ from the *be*-matrices in their diagonal entries. The redistribution of valence electrons by which the starting materials of chemical reactions are converted into their products are represented by *r*-matrices. The fundamental equation of the Dugundji-Ugi model for any reaction $\alpha_1 + \alpha_2 + ..\alpha_n \rightarrow \beta_1 + \beta_2 + ..\beta_m$ is $B + R = E$, where B and E are the *be*-matrices of reactants and products, and R is the reaction matrix. The addition of matrices proceeds entry by entry, that is, $b_{ij} + r_{ij} = e_{ij}$. Since there are no formal negative bond orders or negative numbers of valence electrons, the negative entries of R must be matched by positive entries of B of at least equal values. Within the above restrictions, Dugundji and Ugi proved that *be*- and *r*-matrices forms a free additive abelian group.[11] There are two types of formal reaction generators, RGB and RGR. RGB generators solve the fundamental equation for a given B, while RGR generators solve the equation for a given R. Additionally, constraints can be added while solving either RGB or RGR equations. Example of constraints are maximum number of species and reactions generated, maximum species size, maximum number of reactants per reaction, maximum number of lone electrons, etc.

Table 1. Network Generator Constraints[a]

	maximum value allowed			
	DNG	*RSNG*	*CSNG*	*MCNG*
	Network Constraints			
r	∞ (3)	∞ (3)	∞ (3)	∞ (3)
R	∞ (8)	∞ (8)	∞ (8)	∞ (8)
O_{max}	2 (2)	2 (2)	2 (2)	2 (2)
M_s	n/a	100 000 (20)	100 000 (20)	n/a
M_c	n/a	n/a	100 000 (10 000)	100 000 (10 000)
M_p	n/a	n/a	100 000 (10 000)	100 000 (10 000)
	Species Constraints			
n	∞ (50)	∞ (50)	∞ (50)	1 000 000 (50)
e_s	∞ (2)	∞ (2)	∞ (2)	1 000 000 (2)
e_p	2 (1)	2 (1)	2 (1)	2 (1)

[a] Cf. ref 26 for notations. Values in parentheses are actual values taken for hydrocarbon thermal cracking.

The generator proposed in this paper is of type RGB. Consequently, the generator builds all the products that can be generated from a set of reactants and a set of constraints. The current version of our generator is limited to monomolecular and bimolecular reactions but could easily be extended to more complex reactions. Following Dugundji-Ugi methodology reactants are represented by *be*-matrices. The constraints are entered by the user from the list given in Table 1. The studied chemical process is represented into the form of elementary transitions, which are stored in a library of *r*-matrices. Elementary transitions are the electronic transitions that atoms of the reacting system can undergo. Each elementary transition is composed of two configurations containing all the atoms participating in a reaction before and after the reaction has taken place. Examples of elementary transitions and associated *r*-matrices are given in Figure 1. Fontain and Reitsan[12] have compiled the complete set of elementary transitions that carbon atoms can take in organic

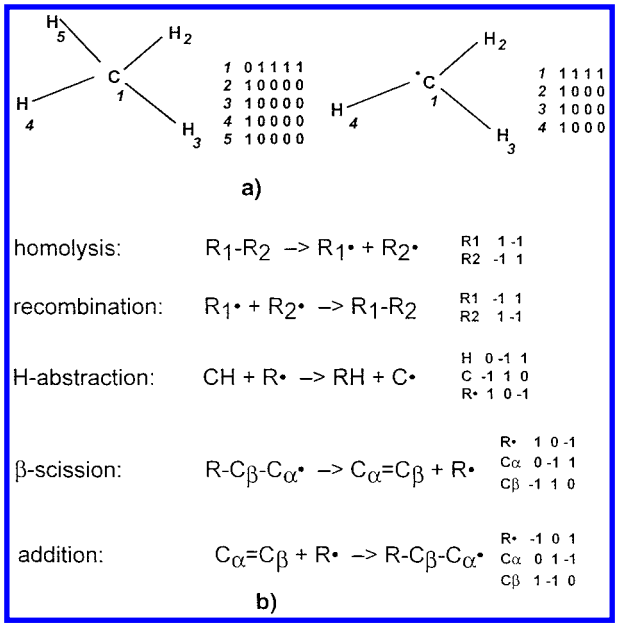

Figure 1. Example of (*be*-) and (*r*-)matrices for elementary transitions of hydrocarbon thermal cracking. (a) (*be*-)matrices for methane and methyl. (b) (*r*-)matrices for the five elementary transitions of hydrocarbon thermal cracking.

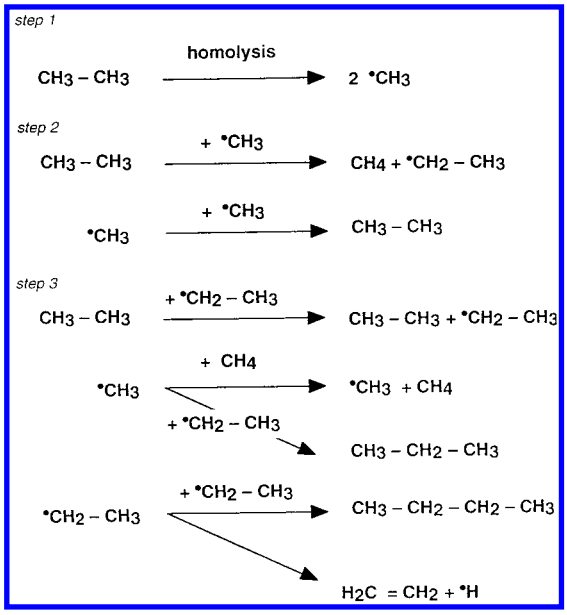

Figure 2. The three first steps of ethane thermal cracking using the five elementary transitions given in Figure 1.

reactions. This set is composed of 324 transitions; however, the set can be greatly reduced depending on the studied process. For instance, it is well-known[13] that all hydrocarbon thermal cracking reactions can be generated from the five elementary transitions listed in Figure 1. As another example Susnow et al.[8] are using 17 transitions to model methane combustion. The algorithms of our generator are given next and are tested for thermal cracking of hydrocarbons in section III.

A. Deterministic Algorithm (DNG). The input of our reaction network generator is a list of reactant species, a list of constraints, and a list of elementary transitions. The output is a network composed of all possible species and reactions that can be created from the input. The algorithm described below is illustrated for ethane thermal cracking in Figure 2. The algorithm starts by computing all the possible species (Ls_1) that can be derived by applying the elementary transitions (*Let*) to the initial species (Ls_0) while respecting the constraints. To prohibit duplication of species, Ls_1 is composed of species that are not already present in Ls_0. When applying elementary transitions, one has to make the distinction between monomolecular and bimolecular reactions. For each monomolecular elementary transition, Ls_1 is composed of all the possible products that can be generated by applying the elementary transition in all possible ways to the species of Ls_0. For each bimolecular reaction, Ls_1 is composed of the products derived by applying the elementary transitions to all possible pairs of species in Ls_0. The list of reactions Lr_1 is updated each time an elementary transition applies to a species in Ls_0. Each reaction is represented by a n-tuples composed of the elementary transition (from *Let*), the reacting species (from Ls_0), and the product species (from Ls_1 or Ls_0). Once the list Ls_1has been computed the algorithm proceeds and compute recursively Ls_2, Ls_3, ..., Ls_i. Note that in order to compute Ls_i (i 1) one has to consider monomolecular reactions only from the set Ls_{i-1} since all monomolecular reactions from the sets Ls_{i-2}, Ls_{i-3}, ... have already been computed in the previous steps. For the same reason, the products of bimolecular reactions in Ls_i are generated for every possible pairs of species having at least one element in Ls_{i-1}. To avoid redundancies of species in the list Ls_0, Ls_1, ..., Ls_i, at any step i 1 a new species is added to Ls_i only if the species is not in $Ls_{i-1} \cup Ls_{i-2} \cup ... \cup Ls_0$. The process halts at any step i, where the corresponding lists of species Ls_i and reaction Lr_i are empty. Since bimolecular reactions can potentially create products of infinite molecular weight, to keep a finite network size one has to set a limit for the maximum species size; let n be this limit. Note that with the exception of polymerization reactions it is reasonable to assume that all species in a given network will be limited in size. In turn, note that if the species have a limited size, then the network generation algorithm is guaranteed to converge. Indeed, the maximum number of species, N, that can be formed is bounded by the total number of molecular structures having a number of atoms raging from 1 to the specified value of n. This latter number is equal to the sum of the numbers of isomers containing 1, 2, ... , n atoms. Finally, note that the number of reactions is also finite and is bounded by N^4 which an upper bound on the number of ways of selecting four species (i.e., two reactants and two products) among N species.

The network generator algorithm given in Chart 1 uses specific data structures to describe species and reactions. A molecular species s is represented by a molecular graph $G(s)$. This graph is in turn represented by a *be*-matrix. An elementary transition (*et*) is represented by two molecular graphs, $G_r(et)$ the electronic configurations of the surrounding atoms before the reaction has taken place, and $G_p(et)$ the electronic configuration after the reaction has taken place. From these molecular graphs one constructs a *r*-matrix which is the difference between the two configurations (i.e. *r*-matrix corresponding to $G_p(et)-G_r(et)$). Finally, a reaction, r, is an n-tuple composed of a elementary transition (*et*), a list of reactant species ($L_r(r)$), a list of product species ($L_p(r)$), and a rate constant ($k(r)$). Additionally, the algorithm maintains several lists of species and reactions. The list of reactants (Ls_0), the list of species (Ls_i) and reactions (Lr_i) created at the current step i, and the list of species generated at the

Chart 1

```
deterministic-network-generator(Ls_0, Let, Ls, Lr)
input:  -Ls_0:  list of initial species
        -Let:   list of elementary transitions
output: -Ls:    list of all species in network
        -Lr:    list of all reactions in network
local:  -Ls_i:  list of species created at
         step i
        -Ls_{i-1}: list of species created at
         step i-1
        -Lr_i:  list of reactions created
         at step i
begin
1.  Ls := Ls_0;  Lr := ∅;  Ls_{i-1} = Ls_0;
2.  do
4.      Ls_i := ∅;  Lr_i = ∅;
5.      generate-species-reactions
                (Ls_{i-1}, Ls, Let, Ls_i, Lr_i)
6.      Ls := Ls ⊎ Ls_i;  Lr := Lr ⊎ Lr_i;
7.      Ls_{i-1} := Ls_i;
8.  until (Ls_i = ∅ and Lr_i = ∅)
end

generate-species-reactions(Ls_1, Ls_2, Let, Ls_i, Lr_i)
input:  -Ls_1, Ls_2:  list of species
        -Let:   list of elementary transitions
output: -Ls_i:  list of species created
         at step i
        -Lr_i:  list of reactions created
         at step i
local:  -L_{G_p}:  list of graphs returned
         by generate-product
begin
1.  For all species s in Ls_1 do
2.      For all reactions et ∈ Let
                with order(et) = 1 do
3.          L_{G_p} := ∅;
4.          generate-product
                (G(s), G_r(et), G_p(et), L_{G_p});
5.          update-species-reactions
                (L_{G_p}, et, s, ∅, Ls_i, Lr_i);
6.      done
7.  done
8.  For all species s_1 ∈ Ls_1 and s_2 ∈ Ls_2 do
9.      For all reactions et ∈ Let
                with order(et) = 2 do
10.         L_{G_p} := ∅;
11.         generate-product
                (G(s_1) ∪ G(s_2), G_r(et), G_p(et), L_{G_p});
12.         update-species-reactions
                (L_{G_p}, et, s_1, s_2, Ls_i, Lr_i);
13.     done
14. done
end

generate-product(G(s), G_r(et), G_p(et), L_{G_p})
input:  -G(s):  molecular graph of species s
        -G_r(et):  molecular graph
         of the reactants of transition et
        -G_p(et):  molecular graph
         of the products of transition et
output: -L_{G_p}:  list of graphs obtained
         after applying et to s

local:  -X:  a graph
        -x:  an atom of G(s)
        -G_x:  a subgraph of G(s) rooted on x

begin
1.  For all atom x ∈ G(s) do
2.      For all G_x ⊆ G(s) s. t.  G_x ≡ G_r(et) do
3.          X := G(s) − G_x + G_p(et);
4.          if (species-constraints(X) = TRUE)
            then L_{G_p} := L_{G_p} ⊎ X fi;
        done
5.  done
end

update-species-reactions(L_{G_p}, et, s_1, s_2, Ls_i, Lr_i)
input:  -L_{G_p}:  list of graphs returned
         by generate-product
        -et:  elementary transition
        -s_1, s_2:  reactant species
output: -Ls_i:  list of species created
         at step i
        -Lr_i:  list of reactions created
         at step i
local:  -L_p:  list of connected graphs
        -k:  rate constant

begin
1.  For all graphs G_p ∈ L_{G_p} do
2.      compute L_p the list of
            connected components of G_p;
3.      Ls_i := Ls_i ⊎ L_p;
4.      k := rate-constant(et, s_1, s_2, L_p);
5.      Lr_i := Lr_i ⊎ (et, s_1, s_2, L_p, k);
6.  done
end
```

previous step (Ls_{i-1}). The operator + adds a new species to any list Ls_i if and only if the species is not already present in Ls_i. Since species are represented by molecular graphs the operator + performs a series of graph isomorphism checks between the species to be inserted and the species already in the list. The molecular graph isomorphism routine used in this study has been published elsewhere.[14] The routine's computational scaling is $O(n^2)$, where n is the number of atoms of the species. The following chart gives the deterministic network generator routines. The functions following and rate-constant are rather trivial and not detailed in Chart 1. The function species-constraints verifies the constraints listed in Table 1, the function returns FALSE if any constraint is not verified and TRUE otherwise. The function rate-constant computes the rate constant of a given reaction using a technique described in section III.

B. Sampling Algorithms. Although the size of the network generated by the deterministic algorithm is finite due the limited size of the species, as we already mentioned the number of species in the network grows exponentially with the with n, the number of atoms of the largest species. In fact, as it will be seen in the last section, it is virtually impossible to generated thermal cracking networks for species larger than heptane. As argued in the Introduction there are several techniques to reduce the network size. We propose in this section three network sampling algorithms

Chart 2

```
random-sampling-network-generator(Ls0, Les, Ls, Lr)
input:  -Ls0:  list of initial species
        -Let:  list of elementary transitions
output: -Ls:   list of all species in network
        -Lr:   list of all reactions in network
local:  -Lsi:  list of species created at
         step i
        -Lsi-1:  list of species created at
         step i - 1
        -Lri:  list of reactions created at
         step i
begin
1.  Ls := Ls0; Lr := ∅; Lsi-1 = Ls0;
2.  do forever
4.      Lsi := ∅; Lri = ∅;
5.      generate-species-reactions
                        (Lsi-1, Ls, Let, Lsi, Lri);
        if (Lsi = ∅ and Lri = ∅) then end;
6.      Ls := Ls ⊎ Lsi; Lr := Lr ⊎ Lri;
7.      reduce-mechanism-random(Ls, Lsi);
8.      Lsi-1 := Lsi;
9.  done
end
reduce-mechanism-random(Ls, Lsi)
input:  -Ls:  list of species
        -Lsi:  list of species
         created at step i
output: -Ls:  reduced list of species
        -Lsi:  reduced list of species
         created at step i
const.: -Ms:  maximum number of species
         allowed in Ls
begin
1.  While (|Ls| > Ms) do
2.        select s in Ls at random;
3.        Ls := Ls - s;
4.        if s ∈ Lsi then Lsi := Lsi - s;
5.  done
end
```

based on the detailed reduction idea: random-sampling, concentration-sampling, and Monte Carlo sampling.

(i) Random-sampling algorithm (RSNG). The algorithm is identical to the deterministic network generator with the exception that at each step i the resulting species list Ls is reduced to a size equal to a predefined number, M_s. The reduction is done at random, that is, species are removed at random from Ls until the size of Ls is below M_s. The algorithm is given in Chart 2. The subroutine generate-species-reactions is given in Chart 1.

(ii) Concentration-sampling algorithm (CSNG). With the concentration-sampling algorithm, the reduction of the network is not performed at random but based on the species concentrations. Furthermore, at each step i species are removed from Ls_i and not Ls as with the previous algorithm. Hence, in the present case, the total number of network species is not limited, but the number of species generated at each step is bounded by the predefined value M_s. The main assumption of this method is that if a species created at any given step i has low concentration values over the reaction time, this species will generate products with concentrations at most equal to twice that low value. Therefore, removing low concentration species from the list Ls_i should have a negligible effect on the final product distribution. Note that this assumption is valid only if the species are consumed during the reactions and do not act as catalysts. Indeed if a species is a catalyst even with low concentration, this species could have a relatively large impact on the final product distribution. Hence, catalytic species must be identified prior using this chart. Note that for thermal cracking reactions all the species participating in the reactions are consumed (cf. Figure 1), and therefore, there is no need to identify catalytic species. Using the above concentration assumption, the algorithm works as follows. At each step i the deterministic routine generate-species-reaction is run to compute the list of new species. At this point the network is composed of all species in $Ls_0 \cup ... \cup Ls_i$ and associated reactions. Although the network may not be complete, since the reaction rates are calculated on-the-fly (cf. section III) the time evolution of the species concentrations can be computed by solving the system of differential equations associated with the partial network. In the present algorithm, we use the Monte Carlo Gillespie[15] (MC-Gillespie) technique to solve the system. The MC-Gillespie technique monitors the number of particles for each species versus time. The initial number of particles of the reactants are computed from their initial concentrations (given by the user) and the initial number, M_p, of particles in the system. In the present paper, we are using the MC-Gillespie technique at constant volume V, which is calculated from the initial number of particles and the particle density (both user inputs). The MC-Gillespie technique is an exact method for numerical integration of the time evolution of any spatially homogeneous mixture of molecular species that interact through a specified set of coupled chemical reaction channels. The technique is based on a fundamental equation giving the probability at time t that the next reaction in V will occur in the differential time interval $[t + \tau, t + \tau + d\tau]$ and will be an r_μ reaction

$$P(\tau,\mu)d\tau = P_1(\tau)\, P_2(\mu|\tau) \tag{1}$$

where P_1 is the probability that the next reaction will occur between times $t + \tau$ and $t + \tau + d\tau$

$$P_1(\tau) = ae^{-a\tau}d\tau \tag{2}$$

and P_2 is the probability that the next reaction will be r_μ:

$$P_2(\mu|\tau) = a_\mu/a \tag{3}$$

with

$$a = \sum_\mu a_\mu \tag{4}$$

In eqs 2−4, $a_\mu d\tau$ is the probability, to first order in $d\tau$, that a reaction r_μ will occur in V in the next time interval $d\tau$. The rate constant k_μ is related to a_μ in different ways depending if the reaction is monomolecular or bimolecular. For monomolecular reactions

$$a_\mu = [s]k_\mu \tag{5}$$

where $[s]$ is the number of particles of reactant species s.

Chart 3

```
concentration-sampling-network-
                    generator(Ls_0, Let, Ls, Lr)
input:  -Ls_0:  list of initial species
        -Let:   list of elementary transitions
output: -Ls:    list of all species in network
        -Lr:    list of all reactions in network
local:  -Ls_i:  list of species created at
         step i
        -Ls_{i-1}: list of species created at
         step i - 1
        -Lr_i:  list of reactions created at
         step i
begin
1.  Ls := Ls_0;  Lr := ∅;  Ls_{i-1} = Ls_0;
2.  do forever
4.      Ls_i := ∅;  Lr_i = ∅;
5.      generate-species-reactions
                    (Ls_{i-1}, Ls, Let, Ls_i, Lr_i);
        if (Ls_i = ∅ and Lr_i = ∅) then end;
6.      Ls := Ls ⊎ Ls_i;  Lr := Lr ⊎ Lr_i;
7.      reduce-mechanism-concentration(Ls, Ls_i);
8.      Ls_{i-1} := Ls_i;
9.  done
end
reduce-mechanism-concentration(Ls, Ls_i)
input:  -Ls:    list of species
        -Ls_i:  list of species
         created at step i
output: -Ls:    reduced list of species
        -Ls_i:  reduced list of species
         created at step i
local:  -L[s]:  list of species concentration
        -L[s_max]:  list of species
         maximum concentration
        -n_c:   number of MC steps
        -t:     time
const.: -M_p:   maximum number particles
        -M_c:   maximum number of MC steps
begin
1.  L[s] := ∅;  L[s_max] := ∅;
2.  For all species s ∈ Ls do
3.      if step(s) = 0 then
4.          [s] := assign-initial-number
                    -particles(s, M_p);
5.          [s_max] := [s];
6.      else
7.          [s] := 0;  [s_max] := 0;
8.      fi
9.      L[s] := L[s] ∪ [s];
10.     L[s_max] := L[s_max] ∪ [s_max];
11. done
12. t := 0;  n_c := 0;
13. While n_c < M_c do
14.     t :=MC-Gillespie-step(Ls, L[s], Lr, t);
15.     For all species s ∈ Ls do
16.         if [s] > [s_max] then [s_max] = [s];
17.     done
18.     n_c := n_c + 1;
19. done
20. While (|Ls_i| > M_s) do
21.     find s ∈ Ls_i
          having the lowest [s_max] value
22.     Ls := Ls - s;  Ls_i := Ls_i - s;
23. done
end


MC-Gillespie-step(Ls, L[s], Lr, t)
input:  -Ls:    list of species
        -L[s]:  list of species concentration
        -Lr:    list of reactions
        -t:     time
output: -L[s]:  updated list of species
         concentration
        -t:     time after event occurs

begin
1.  compute time τ of next event using eq.8;
2.  select reaction r in Lr occurring
        at time t + τ using eq.9;
3.  t := t + τ;
4.  For all s ∈ L_r(r) do [s] := [s] - 1 done;
5.  For all s ∈ L_p(r) do [s] := [s] + 1 done;
6.  return t;
end
```

For bimolecular reactions involving two species s_1 and s_2, we have

$$a_\mu = [s_1][s_2]k_\mu/V \tag{6}$$

and for bimolecular reactions involving only one reactant species

$$a_\mu = [s]([s] - 1)k_\mu/2V \tag{7}$$

To integrate eq 1, Gillespie proposes the following chart. First generate a random value τ according to $P_1(\tau)$ and second generate a random integer μ according to $P_2(\mu/\tau)$. In our implementation of the MC-Gillespie technique, the random value τ is computed by simply drawing a random number r_1 from an uniform distribution in the unit interval and taking

$$\tau = (1/a)ln(1/r_1) \tag{8}$$

In turn, the random integer μ is generated by drawing another random number r_2 in the unit interval and by taking μ the integer verifying

$$\sum_{\nu=1}^{\mu-1} a_\nu \le r_2 a \le \sum_{\nu=1}^{\mu} a_\nu \tag{9}$$

In his original paper,[15] Gillespie has proven that expressions 8 and 9 are indeed correct to simulate stochastically the time evolution of homogeneous reactions.

During the MC-Gillespie integration, the *CSNG* algorithm retains for each species present in the network the maximum concentration (i.e., number of particles) reached over the time period the system is integrated. This operation requires the maintainence of two lists: $L[s]$, the list of species concentrations; and $L[s_{max}]$, the list of species maximum concentrations. Species are then sorted by decreasing concentration, the first M_s elements of the sorted list are kept in *Ls* while the other are eliminated. The algorithm is given in the following chart, the routine generate-species-reaction is given in Chart 1. The routine assign-initial-number-particle is not detailed here but returns the initial number of particles for each reactants. This number is simply equal to the product of the maximum number of particles (M_p) by the initial concentration of the reactant. Both numbers are user input.

(iii) MC-sampling algorithm (MCNG). The idea of the MC-sampling algorithm is to perform at the same time both the Monte Carlo integration and the network generation. The advantage of this technique is that there are no assumptions regarding catalyst species. As in the previous case, one starts with an initial reactant concentration given in the form of numbers of particles. The initial total number of particles (M_p) is a user input. At each step, the set of new species is computed using the generate-species-reactions routine, but in the present case these species are generated by applying the elementary transitions only for species having nonzero concentration (i.e., set *Ls** in Chart 4). The concentrations of the new species are set to zero and updated using the MC-Gillespie integration step. The process is iterated until the number of steps exceeds the predefined M_c value. In the following chart, the routine generate-species-reactions is given in Chart 1, and the routine MC-Gillespie-step can be found in Chart 3.

Chart 4

```
MC-sampling-network-generator(Ls0, Les, Ls, Lr)
input:  -Ls0:  list of initial species
        -Let:  list of elementary transitions
output: -Ls:   list of all species in network
        -Lr:   list of all reactions in network
local:  -Lsi:  list of species created at
         step i
        -Lri:  list of reactions created at
         step i
        -L[s]: list of species concentration
        -Ls*:  list of species with
         non-zero concentration
        -Ls*-1: list of species created at
         previous step with non-zero concentration
        -t:    time
const.: -Mp:   maximum number particles
        -Mc:   maximum number of MC steps
begin
1.  Ls := Ls0; L[s] := ∅; Lr = ∅;
2.  For all species s ∈ Ls do
3.      [s] := assign-initial-number
                  -particles(s, Mp);
4.      L[s] := L[s] ∪ [s];
5.  done
6.  t = 0; i = 0; Lsi := ∅; Lri := ∅;
7.  While i < Mc do
8.      Ls* := ∅; Ls*-1 := ∅;
9.      For all s ∈ Ls do
10.         if [s] ≠ 0 then Ls* := Ls* ∪ s;
11.     done
12.     For all s ∈ Lsi do
13.         if [s] ≠ 0 then Ls*-1 := Ls*-1 ∪ s;
14.     done
15.     i := i + 1, Lsi := ∅; Lri := ∅;
16.     generate-species-reactions
17.             (Ls*-1, Ls*, Let, Lsi, Lri);
18.     Ls := Ls ⊎ Lsi; Lr := Lr ⊎ Lri;
19.     For all s ∈ Lsi do [s] := 0 done
20.     t := MC-Gillespie-step(Ls, L[s], Lr, t);
21. done
end
```

III. RATE CALCULATIONS

Rate constants are calculated for each new reaction generated in the routine generate-species-reactions. Because generate-species-reactions is a basic routine called by of all the network generation algorithms, we have implemented a fast techniques to compute the rates. Rate constants are estimated for each reaction using a quantitative structure−property relationship (QSPR) based on the reaction type taken for the elementary transitions listed in Figure 1 and on the Wiener indices of the reactants and products. The Wiener index is simply the sum of the number of bonds between all pairs of atoms in a given molecule or radical.[16] Thus, H_2 has a Wiener coefficient of 1, the methyl radical has a Wiener coefficient of 9, methane has a Wiener coefficient of 16, etc. For large (C_4) hydrocarbon species that exhibit multiple isomeric forms, the Wiener coefficient is lowest for highly branched isomers, and largest for the linear isomer. Thus, the Wiener coefficient reflects the extent of branching of a given C_xH_{2x+1} radical stoichiometry. For further details on the characteristics of the Wiener index the reader is referred to the article of Bonchev and Trinajstic.[17] Since the stability of hydrocarbon radicals decreases with branching[18] in the order $R_3C > R_2CH > RCH_2 > CH_3$, one would expect the reactivity of the radicals to increase in the same order. Putting these two facts together leads to the conclusion that the Wiener coefficient should correlate with radical activity, with higher Wiener coefficients indicating higher reactivity for a given C_xH_{2x+1} radical composition.

The QSPR used here is based on experimental kinetic data taken from Allara and Shaw[19] for each of the reactions given in Figure 1. For a given reaction type, the preexponential factor is taken as the average of the measured preexponential factors for each experiment reported in Allara and Shaw. The activation energy is fitted to the expression

$$E_a = E_0 + \alpha\Sigma W_r + \beta\Sigma W_p \qquad (10)$$

where E_0, α, and β are fitting parameters and ΣW_r and ΣW_p are the sums of the Wiener coefficients of the reactants and products, respectively. The form of eq 10 was chosen rather

Table 2. Rate Parameters for Hydrocarbon Thermal Cracking Reactions[a]

reaction	log A (s^{-1})	E_0 (kcal)	α	β
bond homolysis	16.802	86.950 ± 0.65	−0.02199 ± 0.003	0.02890 ± 0.007
β-scission	13.392	35.144 ± 0.65	−0.01736 ± 0.002	0.02468 ± 0.004
H addition to olefins	9.924	3.252 ± 0.66	0.07558 ± 0.05	−0.07045 ± 0.04
C_xH_{2x+1} addition to olefins	7.893	7.826 ± 0.22	−0.00246 ± 0.0002	0.00060 ± 0.0001
H abstraction from alkanes by C_xH_{2x+1} radicals to form alkanes	8.365	11.560 ± 0.43	−0.02767 ± 0.01	0.02780 ± 0.009
H abstraction from alkanes by H radicals to form alkanes	10.986	5.481 ± 0.53	0.34390 ± 0.12	−0.39190 ± 0.15
H abstraction from H_2 by C_xH_{2x+1} radicals to form alkanes	9.323	9.853 ± 0.97	−0.29900 ± 0.15	0.28200 ± 0.13
radical recombination	9.843	0	0	0

[a] Cf. eq 10 for notations.

than the more general form

$$E_a = E_0 + \sum_{i=1}^{R} \alpha_i W_{ri} + \sum_{i=1}^{P} \beta_i W_{pi} \qquad (11)$$

where the α_i's and β_i's are now individual coefficients for each reactant or product. This choice was made to avoid the necessity of sorting the reactants and products to ensure that the appropriate coefficient is applied. For example, a hydrogen transfer reaction where reactant 1 is an alkane and reactant 2 is a radical would require that the algorithm differentiate between the radical and the alkane to make sure that W_1 is multiplied by α_1 rather than α_2. A similar consideration applies to the products. Coding the sorting was deemed unjustified given the uncertainties in the available experimental data and the resulting unavoidable uncertainties in the QSPR estimations (see below). Note that choosing eq 10 vs eq 11 has no effect on the overall computational complexity of our algorithms. Because the number of reactants and products of any reaction is always physically limited, the computational cost of sorting reactants and products is constant and should be ignored in complexity evaluations.

Results from fitting eq 10 to each reaction are given in Table 2. Note that for two of the reactions (addition and hydrogen abstraction) the data are divided into multiple sets rather than being fit as a whole. This division is based on inspection of the data in Allara and Shaw, which reveals that for these two reactions the rate preexponential factors and/or activation energies clearly vary with reactant type. For example, the kinetic parameters of hydrogen abstraction from H_2 clearly differ from those for hydrogen abstraction from an alkane. Consequently Table 2 lists eight different reactions vs only five in Figure 1.

Figure 3 shows a plot of the predicted rate constants for each reaction vs the measured rate constants. In general, the predicted rate constants are within about a half an order of magnitude of the measured values. Unfortunately, the available experimental data also displays variations of this magnitude, so better accuracy cannot be expected from the QSPR. It is nevertheless interesting to note that for the addition reactions and hydrogen transfer from alkanes to alkyl radicals, Figure 3 shows very little variation in the predicted rate constants. This result suggests that the Wiener coefficient is not a good structural parameter for developing a QSPR for these reactions. While the Wiener index decreases with branching it increases with molecular size. Therefore, the summation in eq 10 may mix species with opposite trends of changing Wiener, thus predetermining the low sensitivity of the methods. Subsequent studies should make use of other

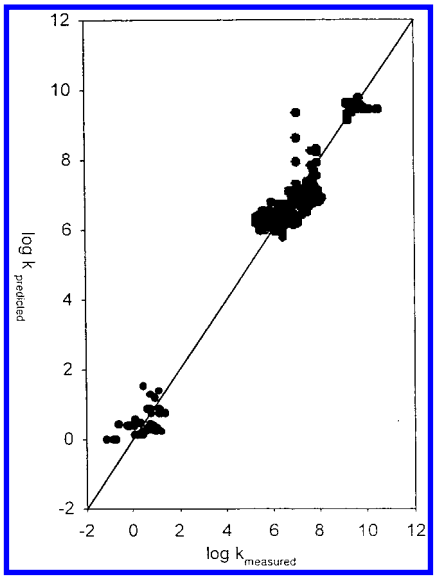

Figure 3. Predicted vs experimental rate constants for all reactions occurring during thermal cracking. Inset shows data for only alkyl addition (black dots), hydrogen transfer tp alkyl groups (white dots), and hydrogen addition (triangles) reactions.

indices such as those surveyed in Todeschini and Consonni's book[20] or utilize more rigorous technique such as linear free energy relationships, semiempirical calculations, or even quantum computations.

Despite these limitations, we believe the QSPR presented here is still accurate enough to demonstrate the performance and scaling of our algorithm and allows for qualitative predictions of product distributions. Let us recall that the goal of this paper is not to present a new technique to compute rate constants but to propose an efficient stochastic network generator that is as accurate as deterministic network generators.

IV. THEORETICAL COMPUTATIONAL COMPLEXITY

In this section we analyze the upper bound theoretical time-complexity of the above algorithms and prove that the three sampling algorithms can be run in polynomial time. Theoretical complexity results are summarized in Tables 3 and 4. The reader should refer to ref 26 for the notations used in this section. Practical results, i.e., running time complexity, are given in the next section.

For hydrocarbon cracking reactions, the maximum number of atoms for each elementary transition is $r \leq 3$ and the number of elementary transitions is $R = 8$ (cf. Figure 1 and Table 2). As a general rule r and R are always bounded and will be taken as constant in our complexity calculations.

Table 3. Number of Species: Upper Bounds[a]

	DNG	*RSNG*	*CSNG*	*MCNG*
$\|Ls_i\|$	$O(N_{0,\,i-1}N_{i-1}n^2)$	$M_s^2O(n^2)$	$M_s^2O(n^3)$	$M_p^2O(n^2)$
$\|Ls_{0,\,i-1}\|$	$O(N_{0,\,i-1})$	M_s	iM_s	$iM_p^2O(n^2)$
$\|Ls\|$	$O(N)$	M_s	$M_sO(n)$	$M_cM_p^2O(n^2)$

[a] Cf. ref 26 for terminology.

We first evaluate the time-complexity of the procedure generate-product. For each atom x of the graph $G(s)$, a series of subgraphs G_x are constructed and tested for isomorphism with $G_r(et)$. We recall that et and $G_r(et)$ contain at most r atoms and the distance between any pair of atoms in $G_r(et)$ is at most $r-1$. Consequently, G_x contains at most all the atoms in $G(s)$ that are at distance $r-1$ from x. The maximum number of atoms in G_x is bounded by $k(k-1)^{r-1}$ where k is the maximum valence allowed (note that $k=4$ for hydrocarbons). Now, the number of graphs G_x containing exactly r atoms is equal to the number of ways of selecting $r-1$ atoms (the root, x, is fixed) in a set of $k(k-1)^{r-1}-1$ atoms, this number is bounded by k^{r2}. Since $G(s)$ comprises at most n atoms, the number of subgraph isomorphisms performed by generate-product is at most $k^{r2}\,n = O(n)$. The algorithm to check subgraph isomorphism simply consist of checking the equality of the *be*-matrices associated to G_x and $G_r(et)$, this can be done in $r^2 = O(1)$ steps since the matrices comprise r rows and r columns. Let us assume that all tries pass the subgraph isomorphism and the constraint tests. In such an instance, $O(n)$ graphs are added to the list L_{Gp}, and consequently, $|L_{Gp}| = O(n)$. Each graph added to L_{Gp} is tested for isomorphism. As already mentioned, the algorithm for graph isomorphism has a complexity of $O(n^2)$.[14] The constraints listed in Table 1 can be checked in $O(n)$ steps, consequently the time-complexity of the generate-product routine is $k^{r2}\,n(n+n^2) = O(n^3)$. Furthermore, the number of graphs returned by generate-product is $|L_{Gp}| \le k^{r2}\,n = O(n)$.

Next, we evaluate the time-complexity of update-species-reactions. As seen above, generate-product may return $O(n)$ graphs, and these graphs in turn may contain at most n connected components (i.e., $|L_p| = n$). Therefore, the maximum number of species and reaction that can be created by update-species-reactions is $|L_{Gp}||L_p| = O(n^2)$. Note that we choose an upper bound for $|Lr_i|$ equal to the upper bound for $|Ls_i|$, since every reaction in Lr_i contains at least one element of Ls_i, and for every element of Ls_i there exists at least one reaction in Lr_i. Note however that we do not necessarily have $|Ls_i| = |Lr_i|$ due to the isomorphism checks performed when inserting elements in Ls_i and Lr_i. The routine update-species-reactions first computes all the connected components of the graphs of L_{Gp}. It is well-known that the connected components of a graph can be found in $O(n)$ steps.[21] The routine then checks isomorphism for all species and reactions created and compute the rate constants of all new reactions. Isomorphism between species can be achieved in $O(n^2)$; isomorphism between reactions can be performed in a constant time (i.e., $O(1)$) since reactions are represented by n-tuple. As seen in the previous section, rate constants are calculated using the Wiener indices of the reactants and products of the reactions. Calculating a Wiener index requires $O(n^2)$ steps for a species containing n atoms.[22] Thus, the overall time-complexity for update-species-reactions is $|L_{Gp}||L_p|(O(n) + O(n^2) + O(1) + O(n^2)) = O(n^4)$.

The routine generate-species-reaction distinguishes monomolecular reactions from bimolecular reactions. For monomolecular reactions all species in $Ls_1 = Ls_{i-1}$ and all elementary transitions Let are sent to generate-product and update-species-reaction. As computed above, the maximum number of species and reactions returned by update-species-reaction scales $O(n^2)$. Therefore, the number of species created through monomolecular reactions and returned by generate-species-reaction is bounded by $|Ls_{i-1}||Let||L_{Gp}||L_p| = N_{i-1}RO(n^2)$. For bimolecular reactions the same operations are performed for all pairs of species between Ls_{i-1} and $Ls_{0,i-1}$, and the number of species created is maximized by $|Ls_{0,\,i-1}||Ls_{i-1}||Let||L_{Gp}||L_p| = N_{0,\,i-1}N_{i-1}R\ O(n^2)$. Consequently, the number of species created at step i is bounded by $|Ls_i| \le N_{i-1}R\,O(n^2) + N_{0,i-1}N_{i-1}R\,O(n^2) = N_{0,i-1}N_{i-1}O(n^2)$. Generate-species-reactions calls the routines generate-product and update-species-reactions for both mono- and bimolecular reactions. The two routines scale respectively $O(n^3)$ and $O(n^4)$. Consequently the time-complexity is $|Ls_{i-1}||Let|(O(n^3) + O(n^4)) = N_{i-1}\,O(n^4)$ for monomolecular reaction and $|Ls_{0,i-1}||Ls_{i-1}||Let| = N_{0,\,i-1}N_{i-1}\,O(n^4)$ for bimolecular reactions. The later being the dominant term the overall time-complexity of generate-species-reaction is $N_{0,i-1}N_{i-1}\,O(n^4)$.

Let us now evaluate the number of iterations in the deterministic-network-generator routine. Let k be the maximum valence allowed. It will take at most kn steps to remove all the bonds in Ls_0 and it will take at most kn steps to create the largest structure comprising n atoms. Therefore, all accessible isomers comprising 1 to n atoms can be created in $2kn$ steps, and the maximum number of iterations of the main routine is bounded by $2kn = O(n)$. Note that we can write $N_{0,i-1} \le 2kn\langle N_i\rangle = N$, where $\langle N_i\rangle$ is the average size of the list Ls_i. Conversely, the upper bound for $|Ls|$ is $N = 2kn\langle N_i\rangle$, and $\langle N_i\rangle = N/(2kn)$. The deterministic-network-generator routine performs $O(n)$ calls to generate-species-reactions, $O(n)$ isomorphism checks between Ls_i and $Ls_{0,\,i}$, and $O(n)$ insertions of Lr_i in $Lr_{0,i}$. Therefore, the upper-bound

Table 4. Computational Time-Complexity: Upper Bounds[a]

	DNG	*RSNG*	*CSNG*	*MCNG*
generate-product	$O(n^3)$	$O(n^3)$	$O(n^3)$	$O(n^3)$
update-species-reactions	$O(n^4)$	$O(n^4)$	$O(n^4)$	$O(n^4)$
generate-species-reactions	$O(N_{0,\,i-1}N_{i-1}n^4)$	$M_s^2O(n^4)$	$M_s^2O(n^5)$	$M_p^2O(n^4)$
MC-Gillespie-step	n/a	n/a	$M_s^2O(n^3)$	$M_cM_p^2O(n^2)$
reduce-mechanism	n/a	$M_s^2O(n^2)$	$(M_s^2 + M_cM_s^2 + M_s^3)O(n^3)$	n/a
full algorithm	$O(N^2n^4)$	$M_s^2O(n^5) + M_s^3O(n^3)$	$M_s^2O(n^6) + (M_c + M_s + 1)O(n^4)$	$M_c^2M_p^4O(n^6)$

[a] Cf. Ref 26 for terminology.

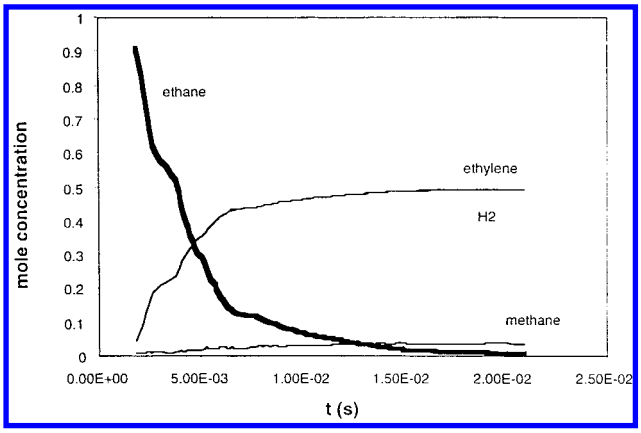

Figure 4. Ethane thermal cracking with *DNG*. $T = 1118$ K, initial ethane concentration = 0.000 545 M.

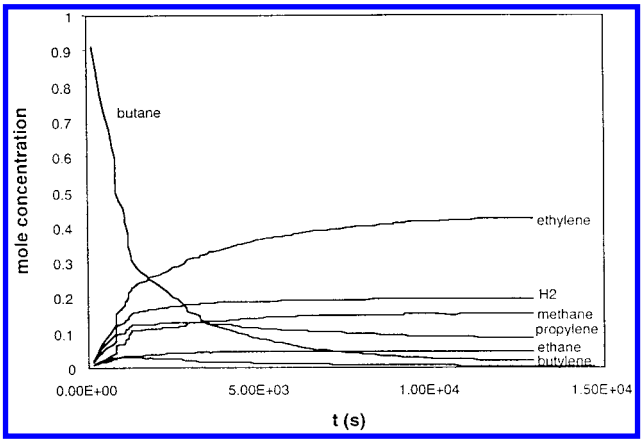

Figure 5. Butane thermal cracking with *DNG*. $T = 863$ K, initial butane concentration = 0.001 M.

time-complexity is $n(N_{0,i}N_{i-1}\ O(n^4) + N_{0,i}N_{i-1}\ O(n^2) + N_{0,i}N_{i-1}) \le n(2kn\langle N_i\rangle N/(2kn)\ O(n^4) + 2kn\langle N_i\rangle N/(2kn)\ O(n^2) + 2kn\langle N_i\rangle N/(2kn)) = N/(2kn)N\ O(n^5) + N/(2kn)N\ O(n^3) + N/(2kn)N\ O(n) \le N^2\ O(n^4)/2k + N^2\ O(n^2)/2k + N^2/2k = N^2\ O(n^4)$. This complexity is not polynomial since N is bounded by the number of isomers comprising 1, ... , n atoms, which increases exponentially with n.

We now consider the sampling routines. At any given step i, the random-sampling routine keeps the size of the list $Ls_{0,i-1}$ below a predefined value M_s. Thus, we have $|Ls_{0,i-1}| \le M_s$. The procedure reduce-mechanism-random performs M_s random selections in $Ls_{0,i-1} + Ls_i$. As seen above, we have $|Ls_i| \le N_{i-1}R\ O(n^2) + N_{0,i-1}N_{i-1}R\ O(n^2) \le N_{i-1}R\ O(n^2) + M_sN_{i-1}R\ O(n^2) = M_s^2\ O(n^2)$, since in the present case both $N_{0,i-1}$ and N_{i-1} are bounded by M_s. Note that we also have $|Lr_i| \le M_s^2\ O(n^2)$, since $|Ls_i|$ and $|Lr_i|$ have the same upper bounds. Furthermore, $|Ls_{0,i-1} + Ls_i| \le |Ls_{0,i-1}| + |Ls_i| \le M_s + M_s^2\ O(n^2) = M_s^2\ O(n^2)$. Consequently, M_s random selections in $Ls_{0,i-1} + Ls_i$ will be performed with a time-complexity $M_s^3\ O(n^2)$. Let us recall that the complexity of generate-reactions-species is $N_{0,i-1}N_{i-1}R\ O(n^4)$, which in the present case reduces to $M_sM_sR\ O(n^4) = M_s^2O(n^4)$. Following the same reductions, the overall complexity of the random-sampling-network-generator algorithm is bounded by $n(N_{0,i}N_{i-1}\ O(n^4) + N_{0,i}N_{i-1}\ O(n^2) + N_{0,i}N_{i-1} + M_s^3\ O(n^2) = M_s^2O(n^5) + M_s^3O(n^3)$.

Concentration-sampling differs from random-sampling since the size of the list $Ls_{0,i-1}$ is not restrained. Instead, at

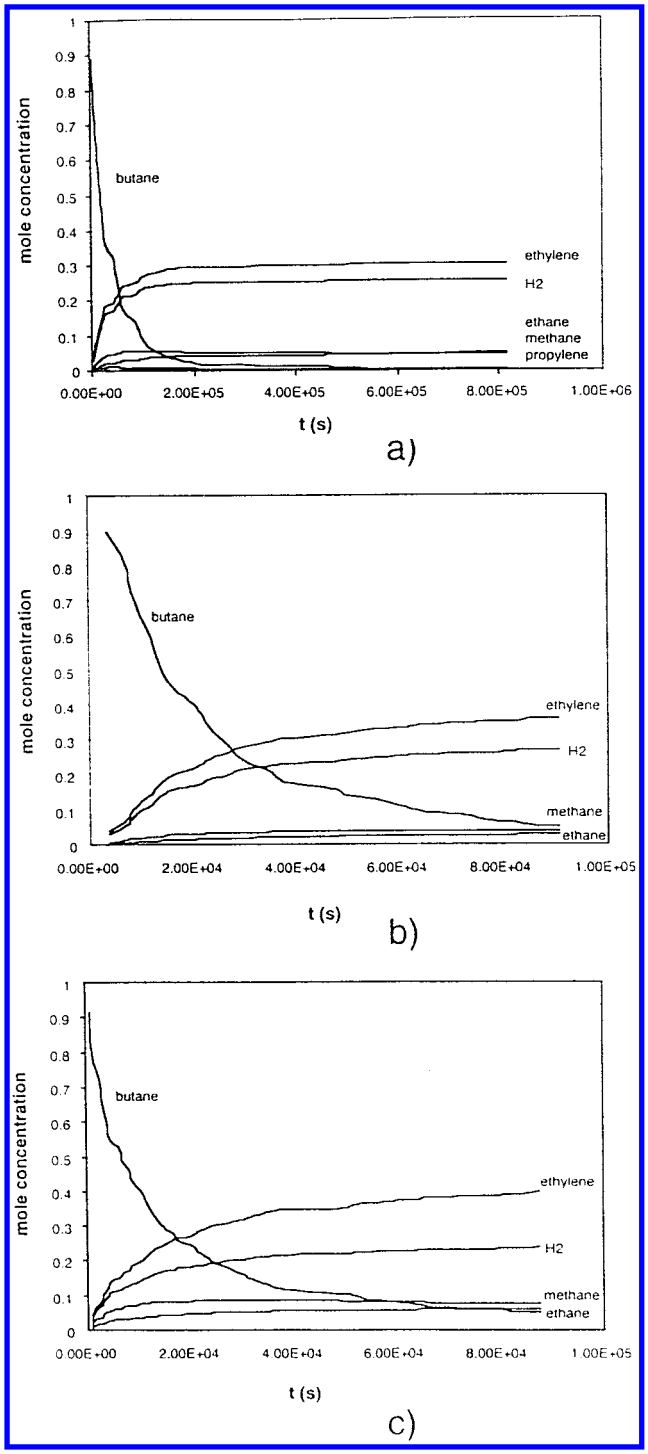

Figure 6. Butane thermal cracking with *RSNG*. $T = 863$ K, initial butane concentration = 0.001 M. (a) *RSNG* with $M_s = 8$, (b) *RSNG* with $M_s = 10$, and (c) *RSNG* with $M_s = 16$.

any given step i, the sizes of the individual lists Ls_0, ... , $Ls_{i-1}$are kept below a predefined value M_s. Thus, we have $|Ls_{0,i-1}| = N_{0,i-1} = iM_s$. We also have $|Ls_i| \le N_{i-1}R\ O(n^2) + N_{0,i-1}N_{i-1}R\ O(n^2) = N_{i-1}R\ O(n^2) + 2kn\langle N_i\rangle N_{i-1}R\ O(n^2) = M_s^2\ O(n^3)$, since in the present case $\langle N_i\rangle$ and $N_{i-1}$are bounded by M_s. Furthermore, $|Ls_{0,i-1} + Ls_i| \le |Ls_{0,i-1}| + |Ls_i| \le 2kn\langle N_i\rangle + M_s^2\ O(n^3) = M_s^2\ O(n^3)$. In turn, the complexity of generate-reactions-species is $N_{0,i-1}N_{i-1}R\ O(n^4)$, which reduces to $2knM_sM_sR\ O(n^4) = M_s^2O(n^5)$.

Concentration-sampling also differs from random-sampling with the reduce-mechanism routine, which calls the MC-

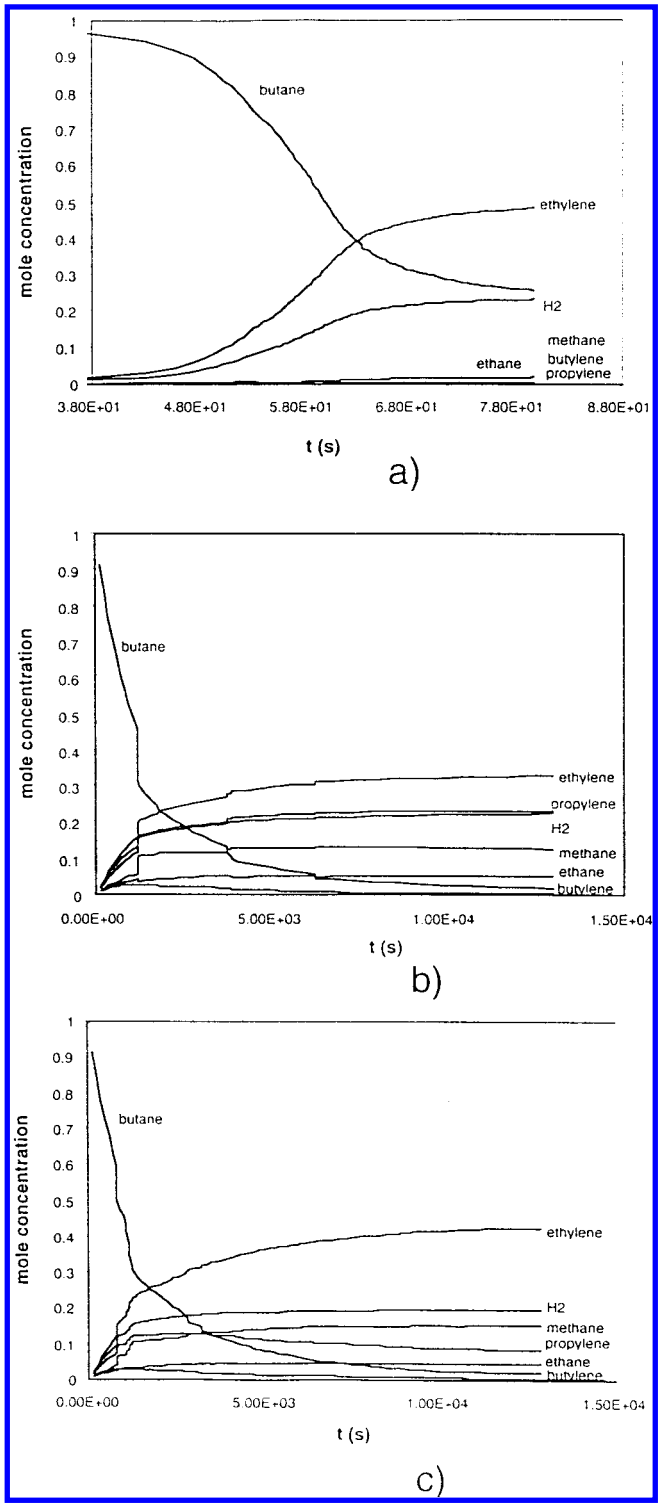

Figure 7. Butane thermal cracking with *CSNG*. $T = 863$ K, initial butane concentration = 0.001 M. (a) *RSNG* with $M_s = 6$, (b) *RSNG* with $M_s = 7$, and (c) *RSNG* with $M_s = 8$.

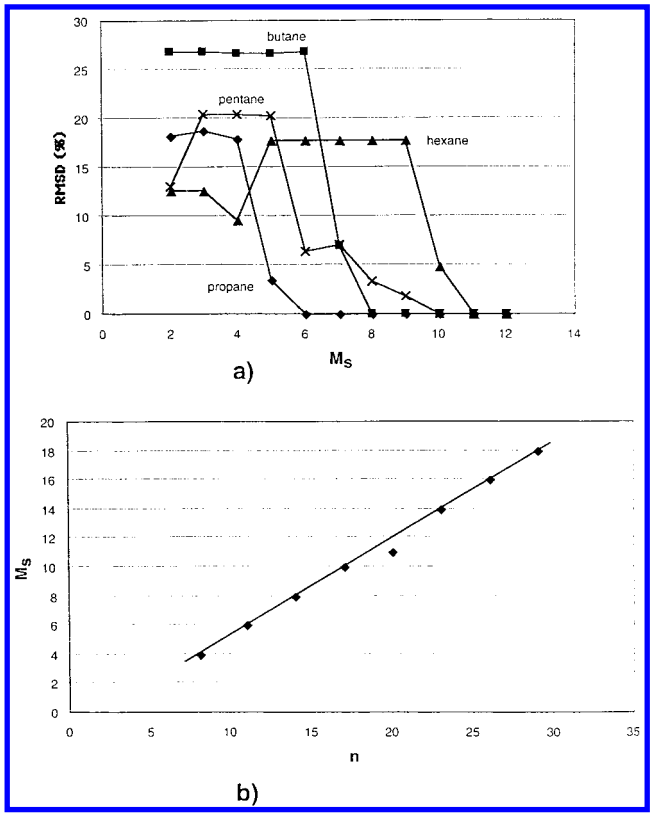

Figure 8. Product distribution RMSD (root-mean-square difference) between *DNG* and *CSNG*. (a) RMSD for individual hydrocabons for different M_s values. (b) M_s vs reactant size (n) for which RMSD = 0. The equation of the straight line is $M_s = 2/3\ (n - 2)$.

Gillespie-step. The MC-Gillespie-step routine first computes the time τ of the next event, then the reaction occurring at time τ, and finally the routine updates the species concentrations. Computing the time of next event is performed using eq 8 and requires $|Lr_i| \leq M_s^2\ O(n^3)$ steps. Selecting the next reaction is also achieved in $|Lr_i| \leq M_s^2\ O(n^3)$ steps. Finally, updating the species concentrations is done in a constant time since there are at most four species (i.e. two reactants and two products) participating to a reaction. The overall time-complexity of MC-Gillespie-step is $M_s^2\ O(n^3)$.

The MC-Gillespie-step is called by the reduce-mechanism-concentration routine. There are three loops in this routine. Clearly, the complexity of the first loop is $|Ls_{0,i-1} + Ls_i| = M_s^2\ O(n^3)$. The second loop calls the Monte Carlo-Gillespie-step routine M_c times, and updates the maximum concentration for each species in $Ls_{0,i-1} + Ls_i$. The MC-Gillespie-step routine has a complexity of $M_s^2\ O(n^3)$. Updating the maximum concentration is achieved in $M_s^2\ O(n^3)$. Therefore the complexity of the second loop is $M_c M_s^2\ O(n^3)$. The third loop searches the M_s maximum concentrations over time of all species in $Ls_{0,i-1} + Ls_i$; its complexity is $M_s^3 O(n^3)$. The overall time-complexity of reduce-mechanism-concentration is therefore $(M_s^2 + M_c M_s^2 + M_s^3)\ O(n^3)$. Thus, the complexity of the concentration-sampling-network-generator algorithm is bounded by $n(N_{0,i}N_{i-1}\ O(n^4) + N_{0,i}N_{i-1}\ O(n^2) + N_{0,i}N_{i-1} + (M_s^2 + M_c M_s^2 + M_s^3)\ O(n^3)) = M_s^2 O(n^6) + (M_c + M_s + 1)M_s^2\ O(n^4)$.

The main difference between MC-sampling-network-generator and the other algorithms is that the number of atoms remains the same, M_p. Hence, the number of species with nonzero concentration is at most M_p (i.e., $|Ls^*| \leq M_p$ and $|L_{-1}^*| \leq M_p$). The routine generate-species-reactions is called for the list Ls_{-1}^* and Ls^* consequently, its complexity is $N^*N^*R\ O(n^4) = M_p^2\ O(n^4)$. Note also that $|Ls_i|$ the number of species returned as well as $|Lr_i|$ the number of reactions are bounded by $|Ls_{-1}^*||Ls^*||Let||L_{Gp}||L_p| \leq M_p^2 R\ O(n^2) = M_p^2 O(n^2)$. The number of species before the first iteration of MC-sampling-network-generator (i.e., the number of reactants) is at most M_p. The number of species after the

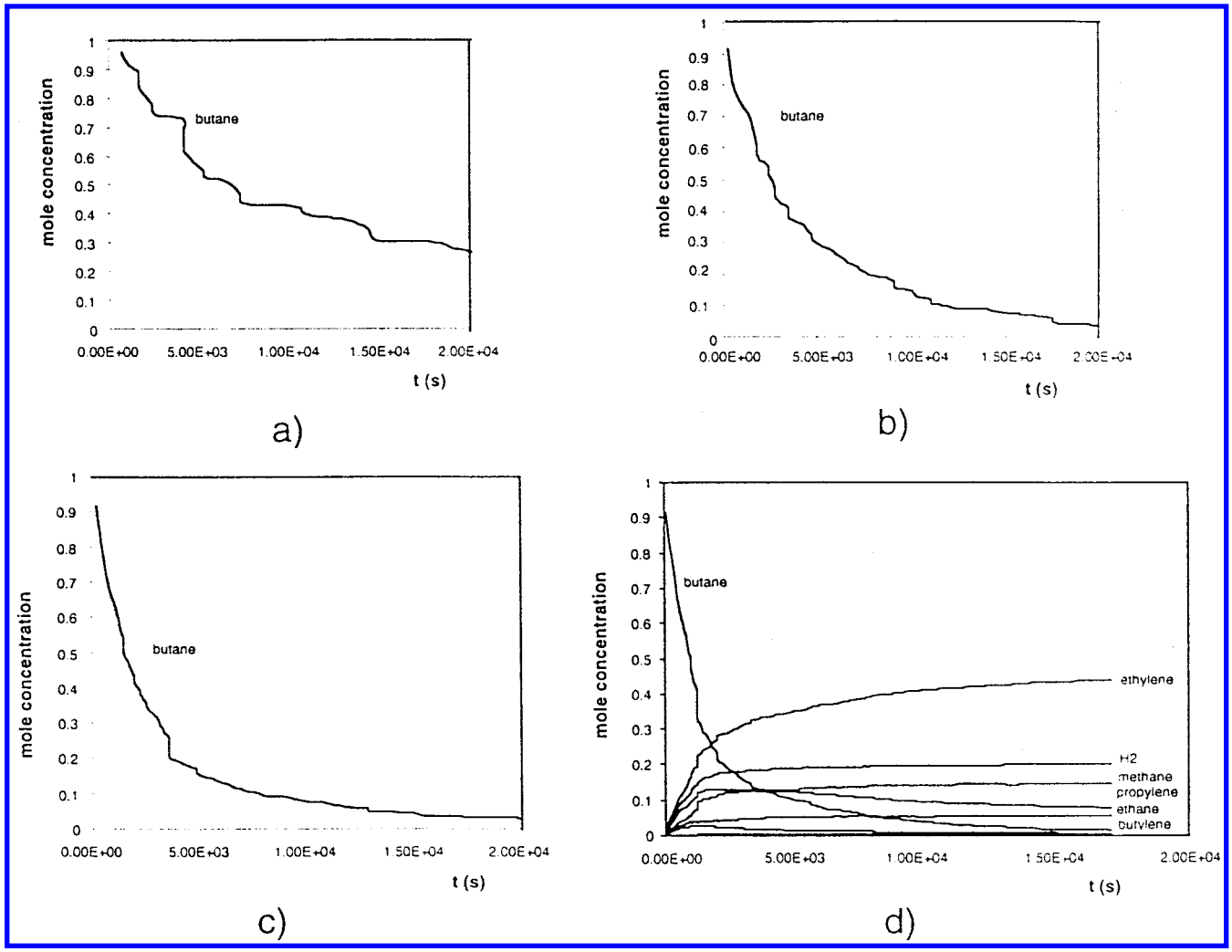

Figure 9. Butane thermal cracking with *MCNG*. $T = 863$ K, initial butane concentration = 0.001 M. (a) $M_p = 100$, (b) $M_p = 500$, (c) $M_p = 2000$, (d) $M_p = 10\,000$ (all major products are shown). In all cases *MCNG* was run until convergence, that is for at most $M_c = 10\,000$ steps.

first iteration is $|Ls_0 + Ls_1| \leq M_p + M_p^2\mathrm{O}(n^2) = M_p^2\mathrm{O}(n^2)$. It can easily be verified that after i iterations the number of species $|Ls_{0,i}|$ will remain bounded by $iM_p^2\mathrm{O}(n^2)$. After M_c iteration we have $|Ls| \leq M_cM_p^2\mathrm{O}(n^2)$ and $|Lr| \leq M_cM_p^2\mathrm{O}(n^2)$ since $|Lr|$ has the same upper bound than $|Ls|$. Hence, the complexity of the isomorphism checks in MC-sampling-network-generator is $|Ls||Ls_i|\ \mathrm{O}(n^2) = M_cM_p^4\mathrm{O}(n^6)$, while the complexity of MC-Gillespie-step is $|Lr| \leq M_cM_p^2\mathrm{O}(n^2)$. Since the number of iteration is equal to M_c, the overall complexity of MC-sampling-network-generator is $M_c^2M_p^4\mathrm{O}(n^6)$.

V. HYDROCARBON THERMAL CRACKING

The goal in this section is to probe how well the sampling algorithms predict the kinetics of hydrocarbon thermal cracking. More precisely, we want to verify if the sampling algorithms can reproduce the results obtained with the *DNG* deterministic network generator. Prior to testing our sampling algorithms it is worth checking how our *DNG* algorithm performs versus experimental results. Although match with experimental data is not our priority here, we want at least to verify that we qualitatively capture the kinetics of hydrocarbon cracking.

(i) *DNG* versus experimental results. Figures 4 and 5 give the product distribution of ethane and butane using the *DNG* algorithm. Comparison of the results with experimental measurements demonstrates that the generated reaction networks and rate constants give only qualitative agreement.

In the case of ethane thermal cracking at 1118 K and an ethane pressure of 38 Torr, our *DNG* technique predicts ethylene and hydrogen in nearly equal amounts as products, with only a small amount of methane formation, regardless of conversion. Experimentally,[23] significant amounts of methane are formed, the ethylene yield falls from 100% to 75% of the hydrogen yield as conversion increases, and small amounts of acetylene and butadiene are formed. The failure of our *DNG* technique to predict the latter two experimental results is entirely due to the absence in our model of reactions that convert ethylene into acetylene and butadiene. Inclusion of these reactions is expected to rectify this inaccuracy. The low methane yield predicted by our simulation is likely due to inaccuracies in the QSPR used to calculate reaction rates, as our simulation generates all of the reactions commonly believed to occur during ethane thermal cracking.

For *n*-butane thermal cracking the agreement between experiment and *DNG* output is also qualitative. While our *DNG* technique predicts the formation of all of the major

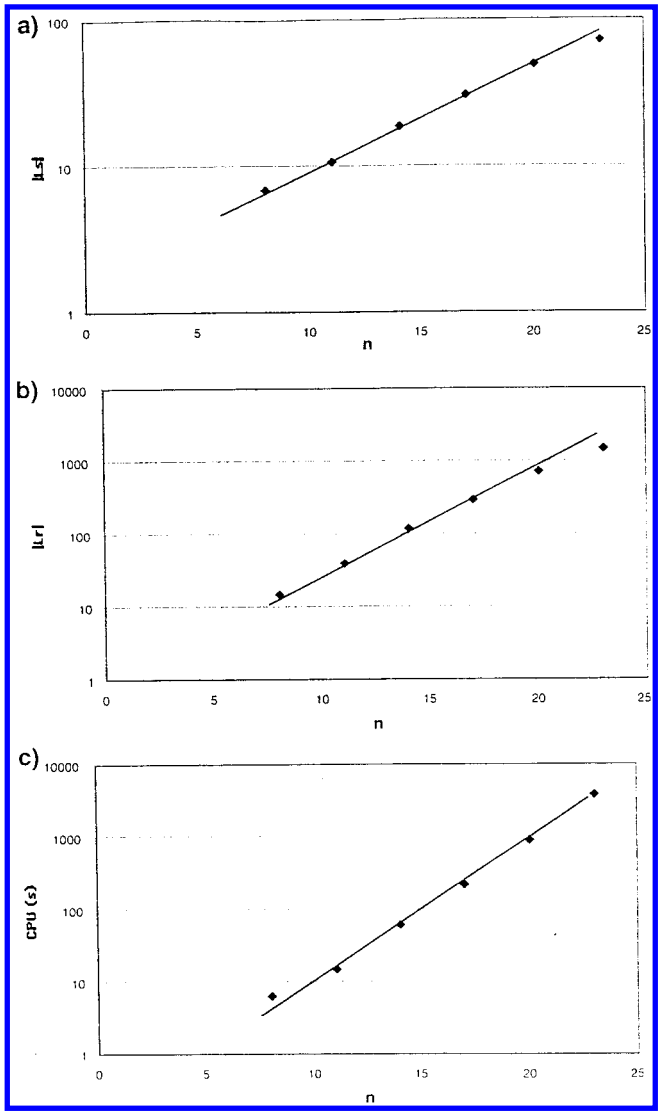

Figure 10. *DNG* computational scaling.

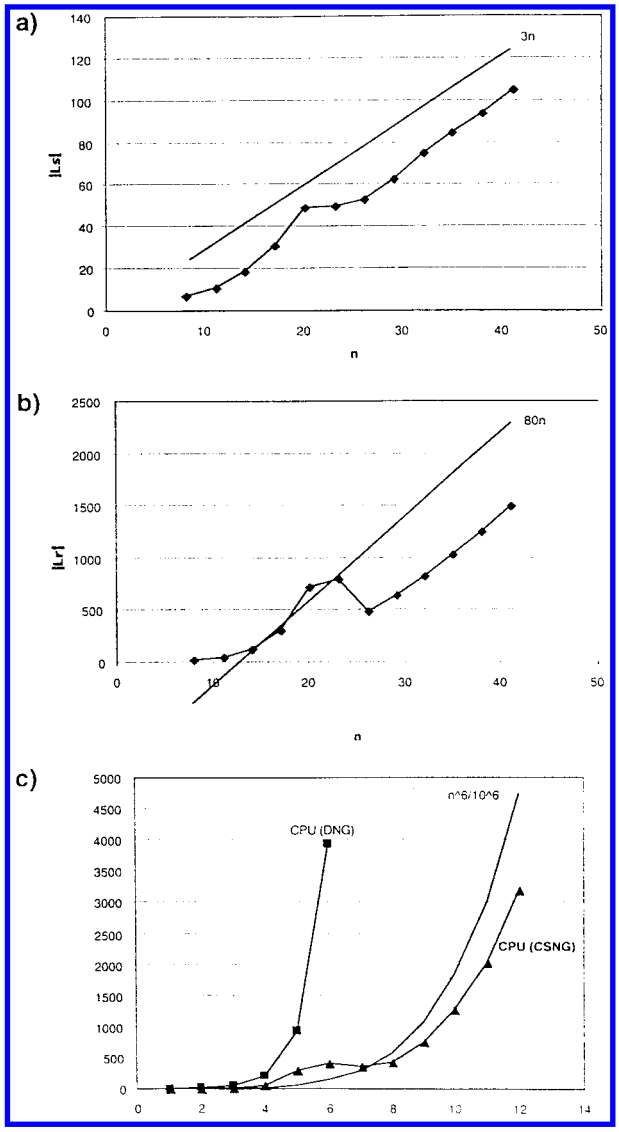

Figure 11. *CSNG* computational scaling for $M_s = 2/3\ (n - 2)$.

products observed experimentally,[24,26] the relative amounts of the products do not agree well. In particular our *DNG* technique underpredicts the amount of methane and overpredicts the amount of hydrogen. As is the case for ethane, the *DNG* algorithm generates all of the important species and reactions for thermal cracking, leading to the conclusion that the discrepancies in product distribution with experiments are the result of inaccuracies in rate parameters predicted from our QSPR.

As noted in Section III, the inacuracies in rate parameters predicted from our QSPR are in large part a reflection of uncertainties in experimental measurements of rate parameters, so that better agreement with experimental product distributions cannot be expected. It should be possible to empirically optimize either the QSPR or individual rate parameters to obtain better agreement with experiments. However, for the purposes of this study, we believe that the qualitative agreement between experiment and simulation, and the fact that the algorithms generate all of the required species and reactions, sufficiently demonstrates the correct performance of the algorithms.

(ii) *RSNG*, *CSNG*, and *MCNG* versus *DNG*. The *RSNG* algorithm was tested with butane cracking for three different M_s values. Let us recall that *RSNG* selects at random M_s species at each generation step disregarding rates and concentration considerations. It is therefore not surprising to find descrepencies between *RSNG* and *DNG* results since intermediate species and even final products may be removed arbitrarily during the selection process. None of the product distributions obtained with $M_s = 8$, 10, and 16 in Figure 6 match the deterministic results. These findings indicate that *RSNG* is not a valid technique to sample reaction mechanism.

Figure 7 gives the product distribution for butane cracking using the *CSNG* algorithm with three different M_s values. Let us recall that *CSNG* selects at each generation step the M_s most abundant species. $M_s = 6$ gives a different product distribution that *DNG*, $M_s = 7$ gives a product distribution close to *DNG* except for propylene which is overpredicted. $M_s = 8$ gives the same product distribution as *DNG*. Not surpringly, we find that for all values $M_s \geq 8$ the products distributions of *DNG* and *CSNG* are indentical. Indeed, according to the *CSNG* assumption (cf. section II) additional species of low concentration have only a minor effect on the final product distribution. $M_s = 8$ is therefore the threshold above which all *CSNG* product distributions are correct. In Figure 8, the above observation is verified for all hydrocarbons up to 24 atoms. As can be seen in Figure 8a, for all tested hydrocarbons a threshold for M_s exists above

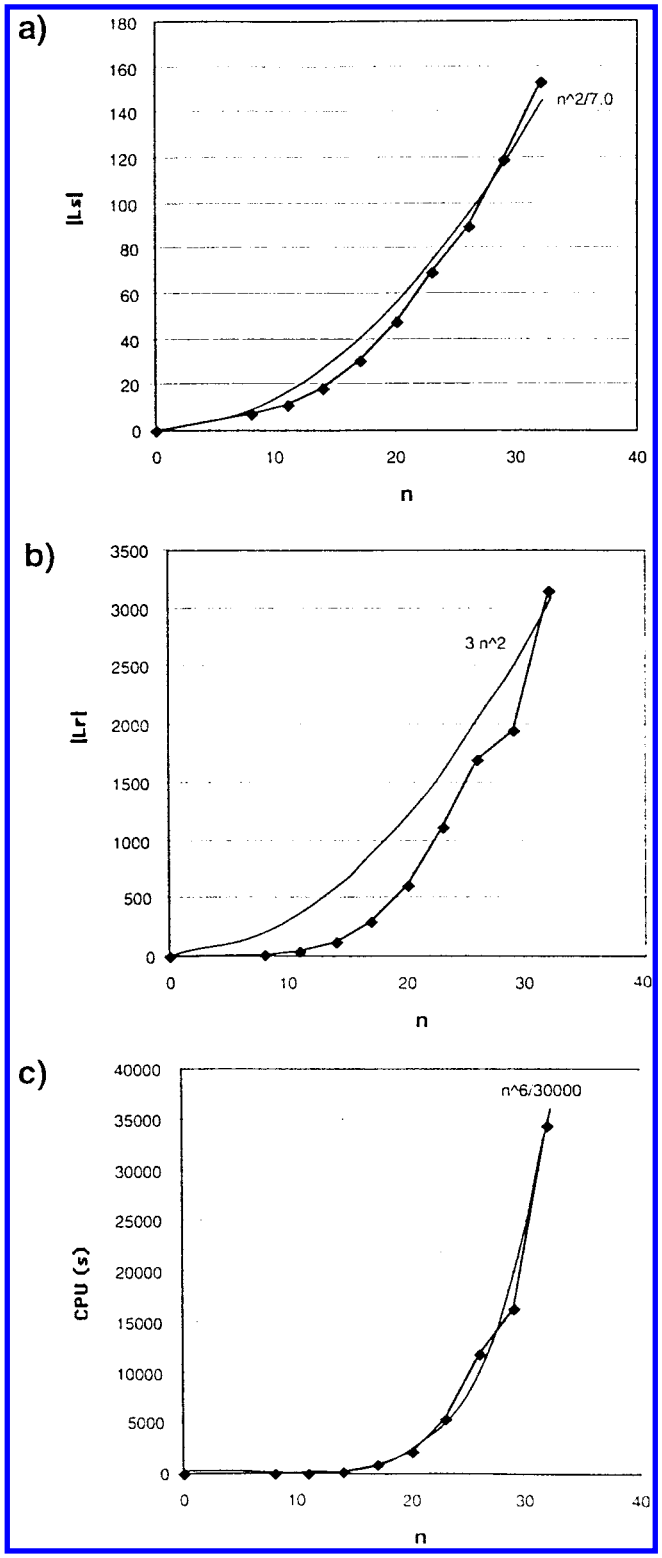

Figure 12. *MCNG* computational scaling for $M_p = 2000$ and $M_c = 10\,000$.

which there is no difference in product distribution between *DNG* and *CSNG*. Furthermore according to Figure 8b, this threshold scales 2/3 $(n - 2)$ where n is the number of atoms of the tested hydrocarbon. All these results lead to the conclusion that *CSNG* can be used to generate reaction network for thermal cracking provided that $M_s \geq 2/3\ (n - 2)$.

For *MCNG* only two parameters can be varied: M_p the initial number of particles and M_c the number of steps on the Monte Carlo integration. As shown in Figure 9, for butane cracking, a perfect match with *DNG* is obtained for $M_p = 1000$ and a fairly good match for $M_p = 2000$. In both cases we have $M_c = 10\,000$. With lower M_p values, butane decomposition is slower, this is due to the fact that not all of the 145 reactions for butane decomposition are used, slower butane decomposition is especially noticeable when the initial number of particle is lower than the number of reactions. $M_p = 2000$ and $M_c = 10\,000$ appear to be the thresholds above which *MCNG* is in good agreement with *DNG*, additional tests (up to up to 24 atoms) indicate that these numbers are independent of the tested hydrocarbon.

(iii) Computational complexity for hydorcarbon thermal cracking. Figures 10−12 give the computational scaling of the *DNG*, *CSNG*, and *MCNG* algorithms for hydrocarbon thermal cracking up to 24 atoms. *RSNG* was not tested for computational scaling since we have shown that it is not an appropriate technique for network generation. In all cases the hydrocarbon results agree well with the theoretical results listed in Tables 3 and 4. With *DNG*, the number of species and CPU time scale exponentially with n (the maximum number of atoms per species allowed). While *CSNG* and *MCNG* give product distribution identical to *DNG*, in both cases, the number of species generated and the CPU time scale polynomially with n.

VI. CONCLUSION

We have introduced here a new technique where network generation and reduction are performed simultaneously. Theoretically, we have proven that the deterministic network generators based on the Dugundji-Ugi methodology have a number of species/reactions and CPU running time that scale exponentially. We have also proven that our concentration sampling and Monte Carlo sampling algorithms scale polynomially while giving identical results than deterministic generators. All our theoretical findings are in agreement with results obtained for hydrocarbon thermal cracking. The computational running time scaling we found with the concentration sampling algorithm is $n^6/10^6$and $n^6/10^4$ with the Monte Carlo sampling algorithm, where n is the maximum species size. Although the polynomial exponent is rather large, these results enables one to study organic reactions on compounds comprising up to 50 atoms in few CPU hours using standard desktop technology. Whereas improvements in the scaling exponent need to be carried out, our sampling algorithms already offer the possibility of generating complex reaction networks such as those studied in combustion and petroleum refining.

We believe to have provided in this paper a solution to the combinatorial explosion that plagues reaction network generation. While published and unpublished heuristics exist to improve time and space complexity of network generation, to the best of our knowledge, this paper presents the first rigorously based attempt. Our sampling algorithms are general and can be used with any chemical process for which the reactants and the elementary transitions are known. Our algorithms are based on sounds physical grounds since networks are reduced using exact equations (eqs 1−8) derived from statistical mechanics. Beyond statistical mechanics and more pragmatically, when a certain amount of reactants, say 10^{23} particles is subjected to a chemical

process, each product of the reactions is a compound for which at least one molecule has been produced. There is no such thing as a 0.5 molecule. Experimentally, each product is a compound that can be detected, in orther words, a compound with a concentration above a threshold imposed by instrumentation precision. Our sampling algorithms deal with a number of particles much smaller than 10^{23} but they are based on the same principles. Our sampling techniques are only limited by our ability to compute rate constants with accuracy. Note, however, that this problem rises when integrating networks regardless of the way the network is generated.

ACKNOWLEDGMENT

We would like to acknowledge the funding provided by the Math Information and Computer Science program of the U.S. Department of Energy under contract DE-AC04-76DP00789.

REFERENCES AND NOTES

(1) Ugi, I.; Bauer, J.; Bley, K.; Dengler, A.; Dietz A.; Fontain, E.; Gruber, B.; Herges, R.; Knauer, M.; Reitsam, K.; Stein, N. Computer-Assisted Solution of Chemical Problems − The Historical Development and the Present State of the Art of a New Discipline of Chemistry. *Angew. Chem., Int. Ed. Engl.* **1993**, *32*, 201−227.

(2) Ugi, I.; Fontain, E.; Bauer, J. Transparent formal methods for reducing the combinatorial abundance of conceivable solutions to a chemical problem. Computer-assisted elucidation of complex mechanism. *Anal. Chim. Acta* **1990**, *235*, 155−161.

(3) Clymans, P. J.; Froment, G. F. Computer-generation of reaction paths and rates equations in the thermal cracking of normal and branched paraffins. *Comput. Chem. Eng.* **1984**, *83*, 137−142.

(4) Hillewaert, L. P.; Dierickx, J. L.; Froment, G. F. Computer Generation of Reaction Schemes and Rates Equations for Thermal Cracking. *AIChE J.* **1988**, *34*, 17−24.

(5) Frenklach, M. Modeling of Large Reaction Systems. In *textitComplex Chemical Reaction Systems, Mathematical Modelling and Simulation*; Warnatz J., Jager W., Eds.; Springer Series in Chemical Physics; Springer-Verlag: Berlin, 1987; Vol. 47, pp 2−16.

(6) Di Maio, F. P.; Lignola, P. G. KING, a KInetic Network Generator. *Chem. Eng. Sci.* **1992**, *47*, 2713−27181.

(7) Broadbelt, L. J.; Stark, S. M.; Klein, M. T. Computer Generated Pyrolysis Modeling: On-the-fly Generation of Species, Reactions, and Rates. *Ind. Eng. Chem. Res.* **1994**, *33*, 790−799.

(8) Susnow, R. G.; Dean, A. M.; Green, W. H.; Peczak. P.; Broadbelt, L. J. Rate-Based Construction of Kinetic Models for Complex Systems. *J. Phys. Chem A* **1997**, *101*, 3731−3740.

(9) Temkin, O. N.; Zeigarnik, A. V.; Bonchev, D. *Chemical Reaction Network. A Graph-Theoretical Approach*; CRC Press: Boca Raton, FL, 1996.

(10) Dugundji, J.; Gillespie, P.; Marquarding, D. Ugi, I. Metric Space and Graphs Representing the LOgical Structure of Chemistry. In *Chemical Applications of Graph Theory*; Balaban, A. T., Ed.; Academic Press: London, 1976; Chapter 6.

(11) Dugundji, J.; Ugi, I. Theory of the be- and r- matrices. *Top. Curr. Chem.* **1973**, *39*, 19−29.

(12) Fontain, E.; Reitsam, K. The Generation of Reaction Network with RAIN. 1. The Reaction Generator. *J. Chem. Inf. Comput. Sci.* **1991**, *31*, 96−101.

(13) Nigam, A.; Klein, M. T. A Mechanism-Oriented Lumping Strategy for Heavy Hydrocarbon Pyrolysis: Imposition of Quantitative Structure−Reactivity Relationship for Pure Components. *IEC Res.* **1993**, *32*, 1297−1303.

(14) Faulon, J.-L. Isomorphism, automorphism-partitioning, and cannonical labeling can be solved of polynomial-time for molecular graph. *J. Chem. Inf. Comput. Sci.* **1998**, *38*, 432−444.

(15) Gillespie, D. T. A General Method for Numerically Simulating the Stochastic Time Evolution of Coupled Chemical Reactions. *J. Comput. Phys.* **1976**, *22*, 403−434.

(16) Wiener, H. Correlation of Heat of Isomerization and Difference in Heat of Vaporization of Isomers, Among Paraffin Hydrocarbons. *J. Am. Chem. Soc.* **1947**, *69*, 2636.

(17) Bonchev, D.; Trinajstic, N. Information-Theory, Distance Matrix, and Molecular Branching. *J. Chem. Phys.* **1977**, *67*, 4517−4533.

(18) Albright. L. F.; Crynes, B. L.; Corcoran, W. H. *Pyrolysis, Theory and Indistrial Practice*; Academic Press: New York, 1983.

(19) Allara, D. L.; Shaw, R. A Compilation of Kinetic Parameters for the Thermal Degradation of *n*-Alkane Molecules. *J. Phys. Chem. Ref. Data* **1980**, *9*, 523−559.

(20) Todeschini, R.; Consonni, V. *Handbook of Molecular Descriptors*; Wiley-VCH: New York, 2000.

(21) Kucera, L. *Combinatorial Algorithm*; Adam Hilger: Bristol, 1989.

(22) Trinajstic, N. *Chemical Graph Theory*, 2nd ed.; CRC Press: Boca Raton, FL, 1992.

(23) McConnell, C. F.; Head, B. D. Pyrolysis of Ethane and Propane. In *Pyrolysis, Theory and Industrial Practice*; Albright, L. F., Crynes, B. L., Corcoran, W. H., Eds.; Academic Press: New York, 1983; pp 25−45.

(24) Corcoran, W. H. Pyrolysis of *n*-Butane. In *Pyrolysis, Theory and Industrial Practice*; Albright, L. F., Crynes, B. L., Corcoran, W. H., Eds.; Academic Press: New York, 1983; pp 47−68.

(25) Rebick, C. Pyrolysis of Heavy Hydrocarbons. In *Pyrolysis, Theory and Indistrial Practice*; Albright, L. F., Crynes, B. L., Corcoran, W. H., Eds.; Academic Press: New York, 1983; pp 69−87.

(26) Abbreviations: *DNG*, deterministic network generator algorithm; *RSNG*, random sampling network generator algorithm; *CSNG*, concentration sampling network generator algorithm; *MCNG*, Monte Carlo sampling network generator *algorithm;* e_s, maximum number of lone electron, or radicals, per *species;* e_p, maximum number of lone electron, or radicals, per atom; $G(s)$, graph associated to species s; $G_r(et)$, graph associated to the reactants of elementary transition et; $G_p(et)$, graph associated to the products of elementary transition et; *Let*, list of all elementary transitions; *Ls*, list of all species produced by algorithm; *Lr*, list of all reactions produced by algorithm; $L(s)$, list of all species concentration; *Ls**, list of all species with nonzero concentration; Ls_0, list of reactants; Ls_i, list of all species produced by algorithm at step i; Lr_i, list of all reactions produced by algorithm at step i; $Ls_{0,i}$, list of all species produced by algorithm up to step i; $Lr_{0,i}$, list of all reactions produced by algorithm up to step i*;* n, maximum number of atom per species; N, maximum number of species produced by algorithm; N^*, number of species with nonzero concentration; N_0, number of reactants; N_i, maximum number of species produced by algorithm at step i; $N_{0,i}$, maximum number of species produced by algorithm up to step i; M_c, maximum number of Monte Carlo steps; M_s, maximum number of species created per generation steps; M_p, maximum number of particles for Monte Carlo integration; O_{max}, maximum reaction order; r, maximum number of atoms per elementary transition; R, maximum number of elementary transitions.

CI000029M