Computers and Chemical Engineering 100 (2017) 1–8

Contents lists available at ScienceDirect

Computers and Chemical Engineering

journal homepage: www.elsevier.com/locate/compchemeng

On-the-fly pruning for rate-based reaction mechanism generation

Kehang Han [a], William H. Green [a,*], Richard H. West [b]

[a] Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, United States
[b] Department of Chemical Engineering, Northeastern University, Boston, MA 02115, United States

ARTICLE INFO

Article history:
Received 12 July 2016
Received in revised form
31 December 2016
Accepted 3 January 2017
Available online 31 January 2017

Keywords:
Memory reduction
Mechanism generation
Pruning

ABSTRACT

The number of possible side reactions and byproduct species grows very rapidly with the size of a chemical mechanism. A memory-efficient algorithm for automated mechanism generation is presented for coping with this combinatorial complexity. The algorithm selects normalized flux as a metric to identify unimportant species during model generation and prunes them with their reactions, without any loss of accuracy. The new algorithm reduces memory requirements for building kinetic models with 200–300 species by about a factor of 4, or for fixed computer hardware makes it possible to create models including about twice as many species as was previously possible. The increased capability opens the possibility of discovering unexplored reaction networks and modeling more complicated reacting systems.

1. Introduction

Detailed kinetic mechanisms, as a bridge between molecular properties and macroscopic phenomena, are increasingly recognized as necessary for better understanding and designing reacting systems to meet economic and environmental needs. In systems with relatively unselective chemistry, such as pyrolysis, combustion, partial oxidation and many polymerizations, tens to even hundreds of thousands of species are present, which makes manual model construction very difficult and time-consuming. Thus, over past decades various automatic kinetic model generation packages have been developed and are increasingly adopted, among which are MAMOX (Ranzi et al., 1995), EXGAS (Battin-Leclerc, 2002), NetGen (Broadbelt et al., 1994), RMG (Song, 2004) and RMG-Py (Gao et al., 2016). The latter, RMG-Py, developed by the Green group at MIT and the West group at Northeastern University, constructs kinetic models by choosing important species based on a flux-ranking strategy.

However, when dealing with complicated reacting systems (e.g., higher carbon number fuels, higher equivalence ratios), automated model generators are often restricted by hardware limitations since numerous species and their reactions quickly fill up the computer memory (Klinke and Broadbelt, 1997). To achieve high fidelity, a very large number of possible reactions, intermediates, and byproducts must be considered when constructing the reaction mechanism. Because the number of possible bimolecular reactions scales as the square of the number of species in the model, the memory usage increases superlinearly (see Fig. 1).

Currently, generating a model by RMG-Py with more than 230 species on a computer with 8 GB RAM leads to a drastic slowdown in performance, as the operating system must constantly swap data between RAM and disk. Although this issue will eventually be relieved by future improvement of RAM size, it usually takes time; historically it took the industry almost 5 years to increase standard RAM size by a factor of 4. Thus it would be very beneficial to improve RMG to reduce the RAM requirements. To deal with this problem, there have been several attempts to develop software that combines model generation and model reduction. Among them is Klinke and Broadbelt's work, which incorporated into their reaction mechanism generation algorithm in NetGen a radical lumping strategy that groups radicals based on their similarity in reactivity, and on-the-fly sensitivity analysis that evaluates the importance of a certain species based on its impact on fluxes of IN (Important and Necessary) species (Klinke and Broadbelt, 1997). This integration allowed NetGen to create more accurate kinetic models.

However, the Klinke and Broadbelt approach cannot be implemented in RMG, due to different model generation strategies: (1) RMG is designed to distinguish all the radicals so that all the thermo-chemistry and kinetic parameters can be calculated from first principles; and (2) edge species in RMG do not react, leaving no impact on the fluxes of core species, which makes sensitivity evaluation ineffective. Consequently, this paper presents an alternative memory-efficient approach for on-the-fly model reduction during mechanism generation. By identifying and pruning unimportant species based on flux analysis in early stages, RMG was able to generate a model of over 400 species without memory shortage.

* Corresponding author.
E-mail address: whgreen@mit.edu (W.H. Green).

http://dx.doi.org/10.1016/j.compchemeng.2017.01.003
0098-1354/

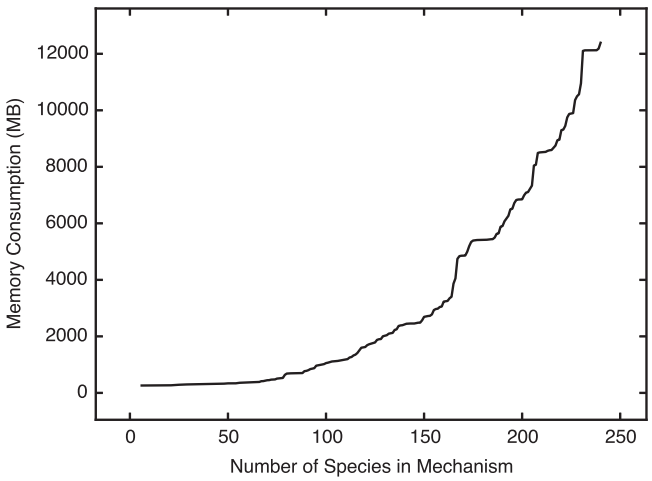

Fig. 1. Memory (RAM) usage by RMG-Py grows super linearly as the reaction mechanism is enlarged to improve fidelity. This example is partial oxidization of natural gas.

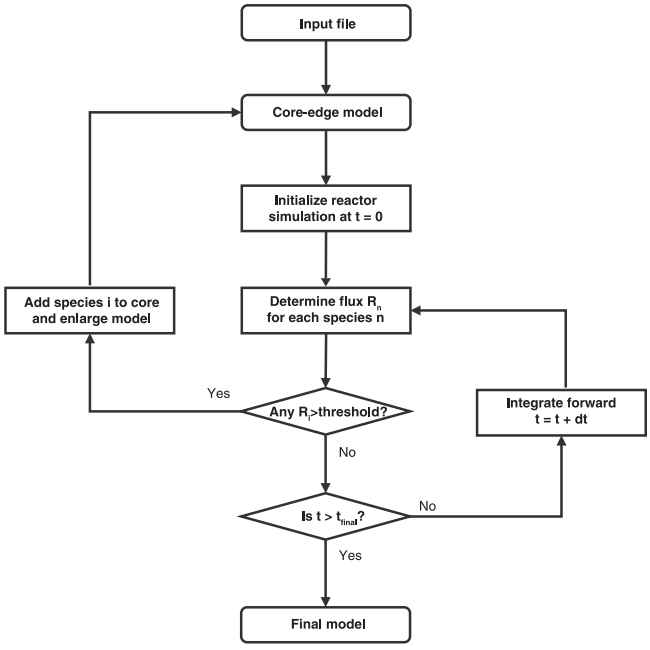

Fig. 2. Model generation workflow of original algorithm.

2. Method

2.1. The original algorithm (no pruning)

Rate-based mechanism generation algorithms such as RMG and RMG-Py work by a "core–edge model" approach (Susnow et al., 1997). The model "core" collects all the important species selected by a rate-based algorithm, while the model "edge" collects all the other species appearing as products of reactions of the core species. The "edge" serves as a species pool for future selections of important species. At each iteration one of the edge species is moved into the "core", and new species are added to the "edge". At the end of model generation, the "core" model will be the final model to be exported.

Typical RMG model generation workflow is illustrated in Fig. 2. User input will be translated to initial core–edge model, which is further transformed into an ODE system. Simulation starts from $t = 0$, along which edge species' fluxes will be monitored at each time point. If an edge species's flux becomes greater than a predefined threshold, that species will be selected to the core, next the new core species will be reacted with other core species so that the model is enlarged. The updated model will trigger a new ODE simulation. This process continues iteratively until the model integrates to the specified final time. All reactions involving only core species are output as the final model.

As Fig. 3 shows in more detail, an RMG iteration starts with a pool of species (species A, B, C in core, D, E, F, and G in edge), and solves the ODE system corresponding to reactions within the set of core species. From the resulting concentrations of core species, reacting fluxes towards each edge species are computed as follows, and then compared to a flux threshold.

$$r_{species_i}(t) = \sum_j \nu^{i,j} r_j(t)$$

where $\nu^{i,j}$ is stoichiometric coefficient of species i in reaction j and r_j is reaction rate for reaction j usually written in Arrhenius form $r_j(t) = k_j T(t)^{n_j} \exp(-(E_a^j/RT(t)))\prod_m c_m^{\nu^{m,j}}(t)$. For instance, at some time the flux towards edge species D is found to be significant, i.e., $r_D(t) > \textit{flux threshold}$, then the computation is halted, and D is moved to the "core". Specifically, the threshold is calculated as $\textit{flux threshold} = R_{char} * \texttt{toleranceMoveToCore}$, where R_{char} is the root sum square of core species fluxes and `toleranceMoveToCore` is specified by the user according to his/her preference of final mechanism accuracy (see more detail on `toleranceMoveToCore` in Section 2.2.1). RMG will later enlarge the core–edge model by exploring reactions between D and other core species. After model "core" and "edge" are updated, the simulator solves the system again from $t = 0$ to the point when next important species is discovered. The whole iterative process terminates when the user-specified goal time/conversion is reached and no additional important species is identified.

2.2. The new algorithm (with pruning)

The goal of RMG is to produce a final kinetic model containing core species and their reactions. However, the number of edge species is much larger than that of core species (typically by several orders of magnitude), so most memory is consumed by the model "edge". Furthermore, among the edge species are many minor species that have little chance to become core species; they can be structurally unstable or difficult to be formed and usually have low $r_{species_i}$. Thus, pruning those minor edge species can be helpful to mitigate RAM limitation. In order to achieve that, pruning module should first identify unimportant edge species and then delete them and their reactions while minimizing the impact on model accuracy.

The pruning module is integrated into the original RMG package illustrated in Fig. 5; the main difference from the original workflow is, after identifying new core species, simulation will continue to final time to figure out `maximum normalized flux` for each edge species:

$$\max_{t \in [0, t_{final}]} \left(\frac{flux_i(t)}{R_{char}(t)} \right)$$

where t_{final} is set by user for simulation termination, see Fig. 2.

Those species with $\max_{t \in [0, t_{final}]} \left(flux_i(t)/R_{char}(t) \right) \leq$ *toleranceKeepInEdge* will be pruned (e.g., in Fig. 4, edge species E and F having relatively small flux are pruned). The simulation will only stop if some edge flux exceeds threshold2 (computed as $\textit{threshold2} = R_{char} * \texttt{toleranceInterruptSimulation}$, see more detail in Section 2.2.2), in which case pruning won't be executed.