

## MIT Open Access Articles

*On-the-Fly Pruning for Rate-Based  
Reaction Mechanism Generation*

The MIT Faculty has made this article openly available. **Please share** how this access benefits you. Your story matters.

**Citation:** Han, Kehang, William H. Green, and Richard H. West. "On-the-fly pruning for rate-based reaction mechanism generation." *Computers & Chemical Engineering*, 100 (May 2017): 1-8.

**As Published:** <http://dx.doi.org/10.1016/j.compchemeng.2017.01.003>

**Publisher:** Elsevier BV

**Persistent URL:** <https://hdl.handle.net/1721.1/123910>

**Version:** Original manuscript: author's manuscript prior to formal peer review

**Terms of use:** Creative Commons Attribution-NonCommercial-NoDerivs License



# On-the-Fly Pruning for Rate-Based Reaction Mechanism Generation

Kehang Han<sup>a</sup>, William H. Green<sup>a,\*</sup>, Richard H. West<sup>b</sup>

<sup>a</sup>*Department of Chemical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139, United States*

<sup>b</sup>*Department of Chemical Engineering, Northeastern University, Boston, MA 02115, United States*

---

## 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  $\sim$  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.

*Keywords:* Memory reduction, Mechanism generation, Pruning

---

## 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 [1], EXGAS [2], NetGen [3], RMG [4] and RMG-Py [5] 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 [6]. To achieve high fidelity,

---

\*Corresponding author

Email address: [whgreen@mit.edu](mailto:whgreen@mit.edu) (William H. Green)

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 Figure 1).

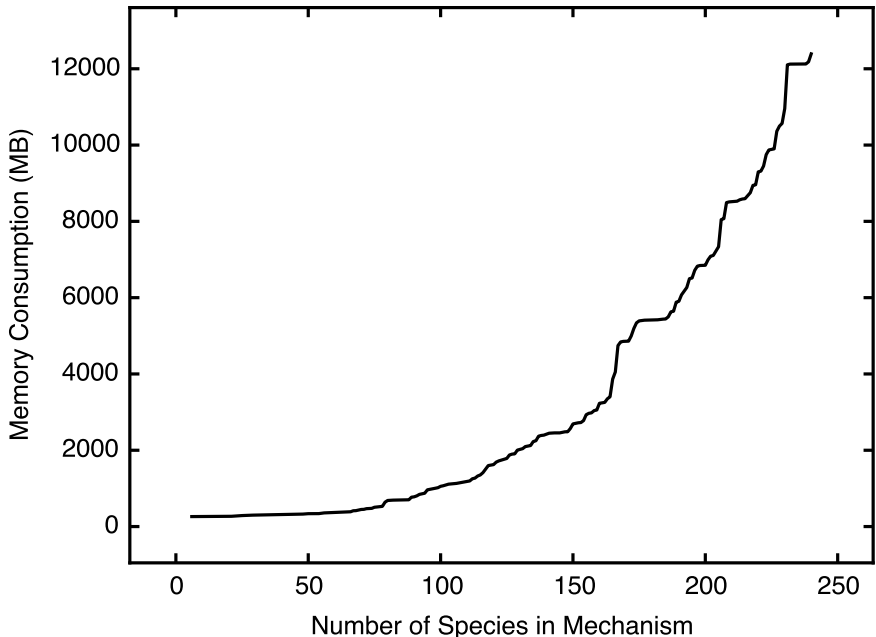


Figure 1: Memory (RAM) usage by RMG-Py grows super linearly as the reaction mechanism is enlarged to improve fidelity. This example is partial oxidization of natural gas.

Currently, generating a model by RMG-Py with more than 230 species on a computer with 8GB 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 [6]. 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.

## 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 [7]. 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 Figure 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 pre-defined threshold, that species will be selected to the core, next the new core species will be reacted with other core species so that model is enlarged. 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 Figure 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  $r_j(t) = k_j T(t)^{n_j} \exp(-\frac{E_a^j}{RT(t)}) \prod_m c_m^{\nu_m^{m,j}}(t)$ . For instance, at some time the flux towards edge species D is found to be significant, i.e.,  $r_D(t) > flux\ threshold$ , then the computation is halted, and D is moved to the “core”. Specifically, the threshold is calculated as  $flux\ threshold = R_{char} * toleranceMoveToCore$ , 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

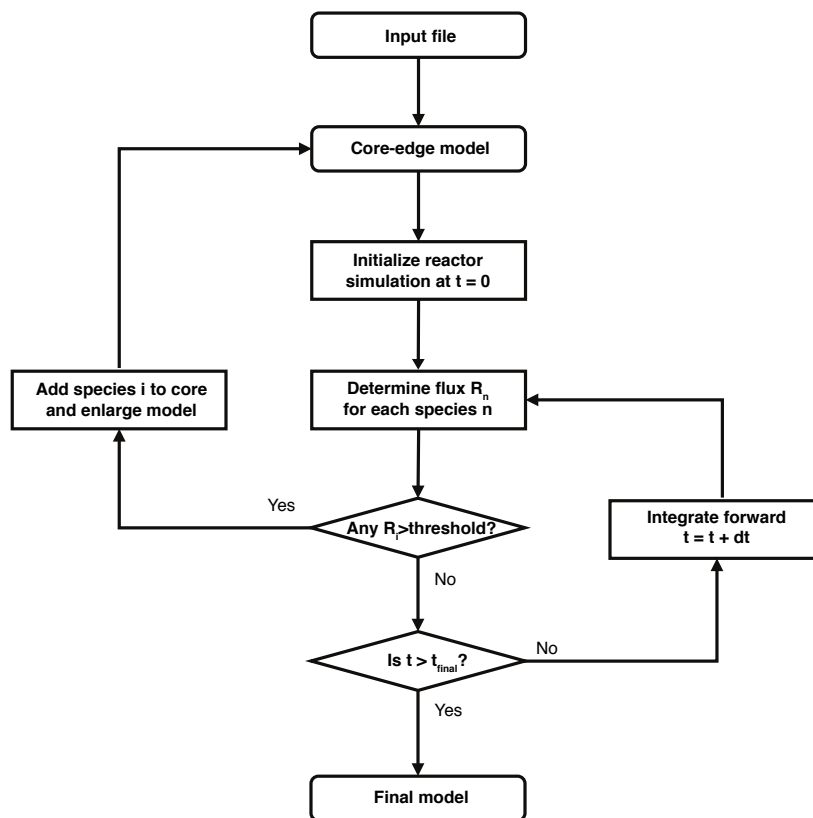


Figure 2: Model generation workflow of original algorithm

mechanism accuracy (see more detail on `toleranceMoveToCore` in 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  $r_{\text{species}_i}$ . 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.

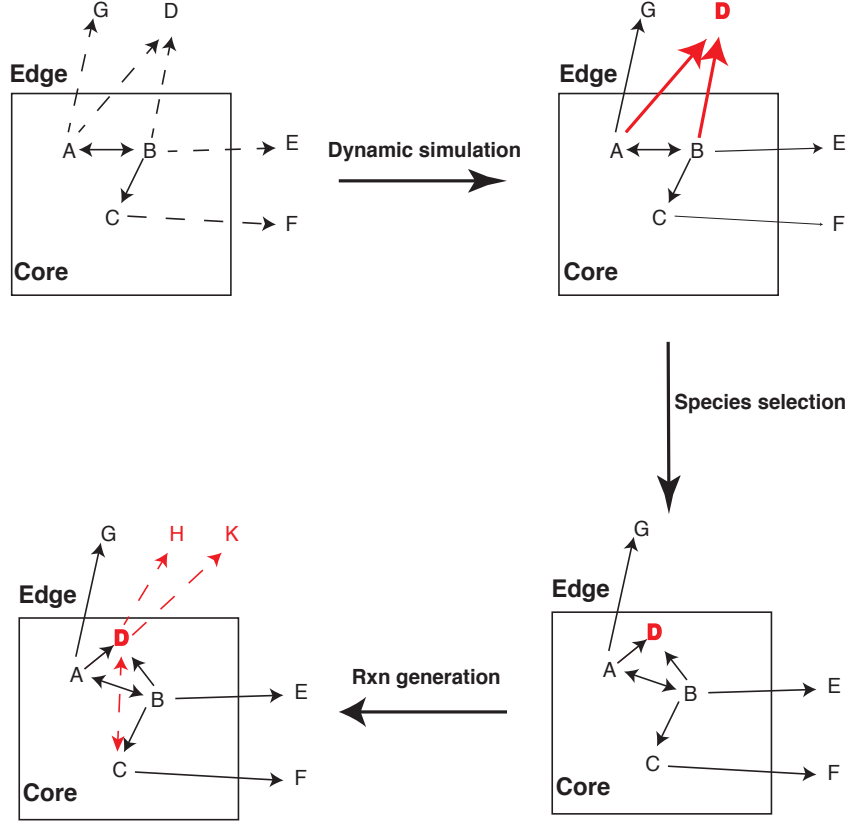


Figure 3: one iteration of model generation by RMG (original algorithm)

The pruning module is integrated into the original RMG package illustrated in Figure 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 Figure 2.

Those species with  $\max_{t \in [0, t_{final}]} \left( \frac{flux_i(t)}{R_{char}(t)} \right) \leq \text{toleranceKeepInEdge}$  will be pruned (e.g. in Figure 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  $threshold2 = R_{char} * \text{toleranceInterruptSimulation}$ , see more detail in 2.2.2), in which case pruning won't be executed.

Four parameters are critical to make pruning functionality work properly.

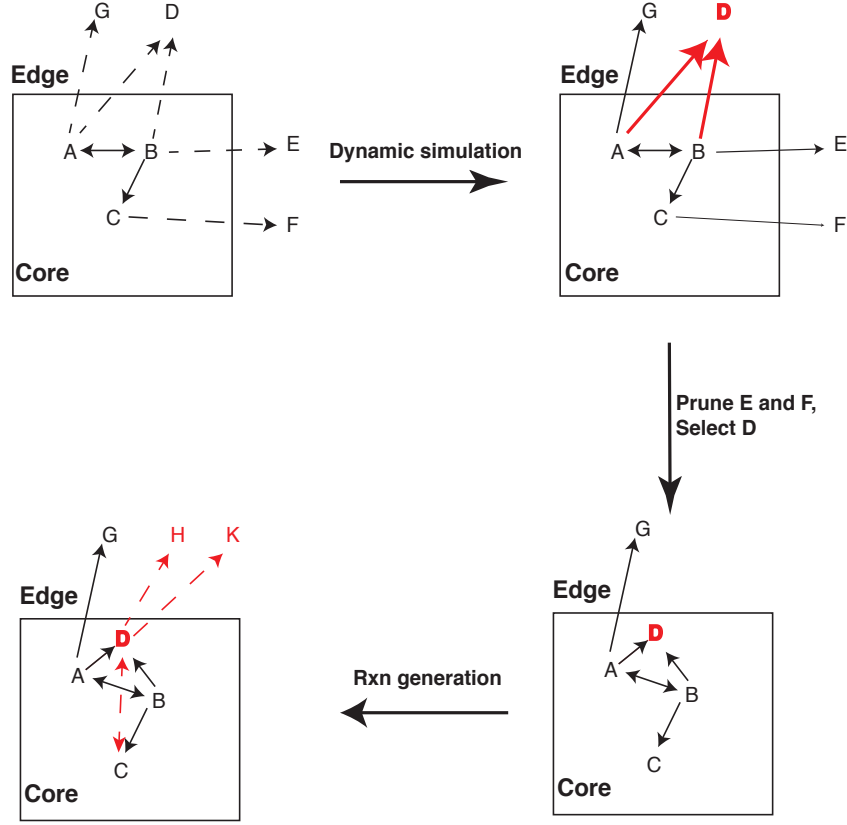


Figure 4: one iteration of model generation by RMG with pruning

### 2.2.1. *toleranceMoveToCore: for threshold1*

This tolerance is inherited from the previous algorithm. Users get final kinetic models with desired accuracy by setting an appropriate `toleranceMoveToCore`. For instance, `toleranceMoveToCore = 0.1` means all the edge species with fluxes  $\geq 10\%$  of  $R_{char}$  will be moved into the final model (model “core”). Normally, lower (tighter) `toleranceMoveToCore` leads to a larger, more detailed final model.

### 2.2.2. *toleranceInterruptSimulation: for threshold2*

RMG has to run a complete dynamic simulation (from time=0 to goal time/conversion) to get  $\max_{t \in [0, t_{final}]} \left( \frac{flux_i(t)}{R_{char}(t)} \right)$  for edge species  $i$ . However, in cases where any of the edge species has an unrealistically high flux, the species is clearly important and must be included in the core first instead of conducting pruning. Thus, `toleranceInterruptSimulation` is defined to decide if a flux is beyond certain limit; kinetic simulation will be halted when some flux is higher than `toleranceInterruptSimulation*`

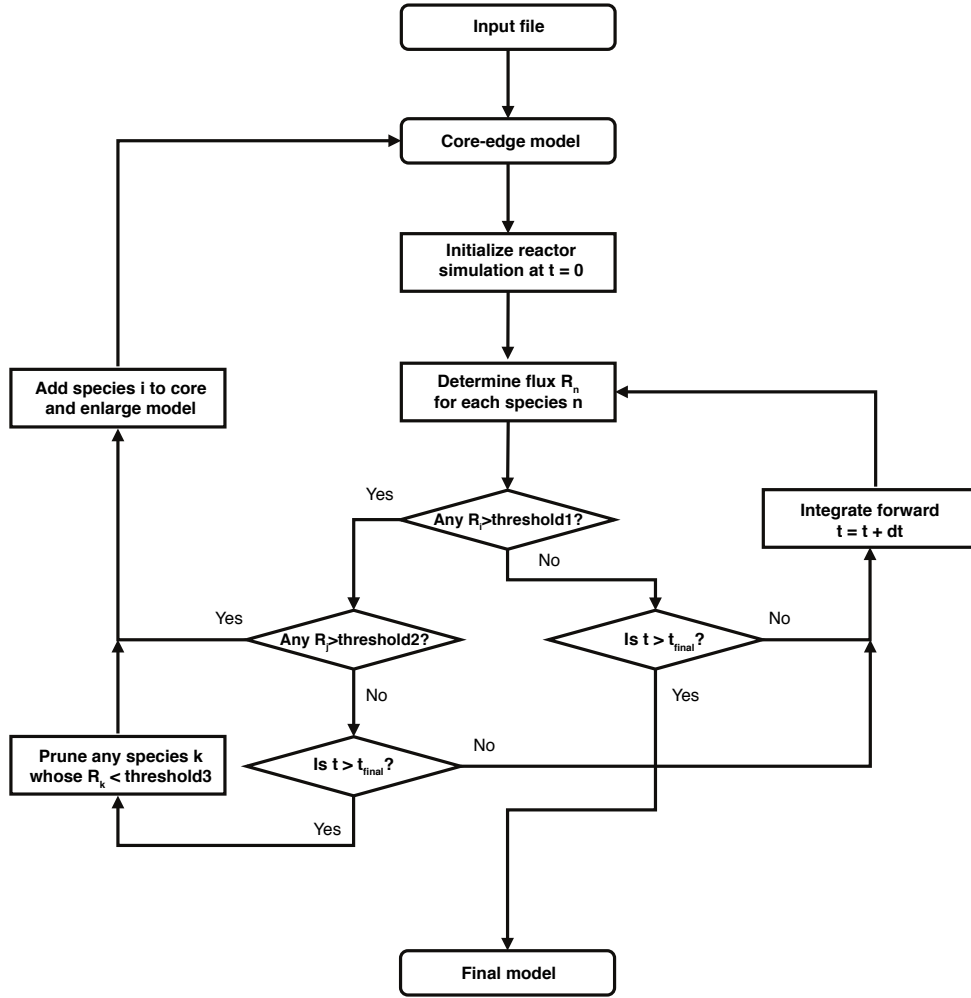


Figure 5: Workflow of pruning algorithm

$R_{char}$ . Small values of `toleranceInterruptSimulation` have the effect of turning off pruning.

### 2.2.3. *toleranceKeepInEdge*: for *threshold3*

As a metric for **importance** of an edge species  $i$ ,  $\max_{t \in [0, t_{final}]} \left( \frac{flux_i(t)}{R_{char}(t)} \right)$  is calculated in each iteration. Any edge species with this metric larger than `toleranceKeepInEdge` at any time  $t$  will be kept in the model edge, i.e., it will not be pruned. Thus, the lower `toleranceKeepInEdge` is, the closer the pruned model will be to a non-pruning model due to fewer edge species being removed.

Naturally, the lower bound for `toleranceKeepInEdge` is 0, indicating no edge species will be deleted (non-pruning scenario); the upper bound is the value of `toleranceMoveToCore`, since `toleranceKeepInEdge` defines the boundary to identify unimportant species while `toleranceMoveToCore` selects the important ones.



#### 2.2.4. *maxEdgeSpecies*

**maxEdgeSpecies** sets an upper bound for total number of edge species. Once it exceeds **maxEdgeSpecies**, RMG will start removing edge species with lowest  $\max_{t \in [0, t_{final}]} \left( \frac{flux_i(t)}{R_{char}(t)} \right)$ , regardless of being higher or lower than **toleranceKeepInEdge**, which will greatly help avoid memory crash but probably harm final model accuracy. It’s always recommended to set **maxEdgeSpecies** as large as the computer can handle. In our experience, with 8 Gb of RAM an appropriate choice for **maxEdgeSpecies** is 100,000.

### 3. Results

Two high-temperature combustion systems were chosen to test the pruning algorithm (see Table 1); one has natural gas (a combination of methane, ethane and propane) as the fuel and O2 as the oxidizer while the other has n-heptane and O2. Both of them are in high equivalence ratios that usually lead to combination of oxidation and pyrolysis, generating significant amounts of intermediates, overflowing RAM with the usual algorithm. However, the two systems have very different chemistries: first system (hereafter, **NG**) goes through a chemical process where small molecules form large ones on the way to soot formation, while the second (hereafter, **C7**) has large molecules decomposing into small ones at the chosen reaction conditions (see Figure 6).

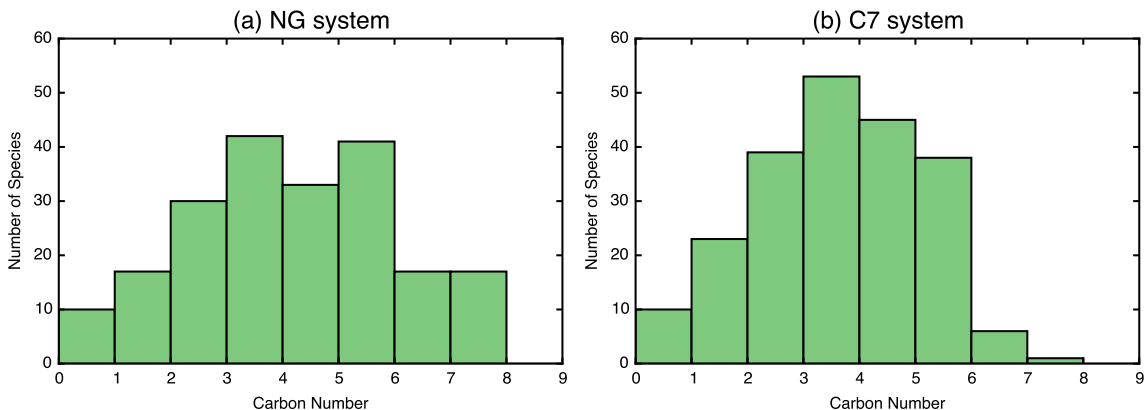


Figure 6: Species size (carbon number) distribution in NG and C7 systems in “core” model at convergence

The new RMG with pruning was able to reduce the quadratic dependence of RAM on core size to nearly linear dependence, as Figure 7 shows.

For various jobs converged to a range of **toleranceMoveToCore** (which generates final models of various sizes), the pruning algorithm not only can generate identical models, but also requires much less memory than the original algorithm does (Figure 8). Pruning saves around 75% of memory for highly complicated systems (low **toleranceMoveToCore**), still generating exactly the same final models as the original algorithm.

system	fuel composition	fuel/ $O_2$ equivalence ratio	T and P
NG	$CH_4$ : 90 mol%	3.4	1400K, 20 atm
	$C_2H_6$ : 8 mol%		
	$C_3H_8$ : 2 mol%		
C7	$C_7H_{16}$ : 100 mol%	11	1400K, 20 atm

Table 1: Specification of test chemistry

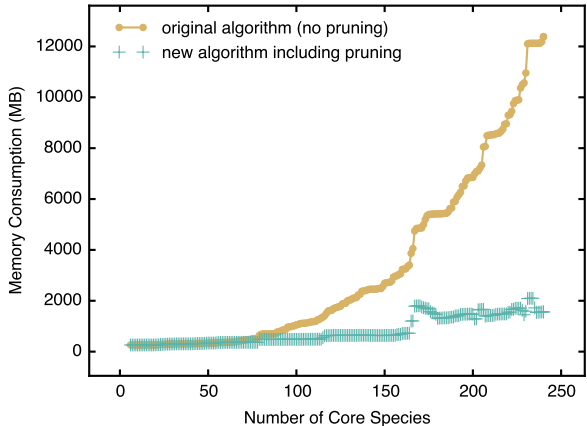


Figure 7: Pruning significantly reduces RAM requirements for the NG system of Figure 1

On the other hand, the pruning algorithm makes it possible to generate large models which the original RMG was not able to; a model with 200 core species can quickly fill up 8Gb RAM using the original algorithm, while the pruning algorithm can easily generate models of 400 core species with memory usage no more than 5Gb, increasing the RMG modeling capability by at least a factor of two (see Table 2).

In order to provide a more concrete example for pruning, phenyl dodecane (hereafter, PDD, see molecule structure Figure 9) thermal decomposition, an heavy oil-to-gas application with which RMG originally fails due to high memory cost, illustrates how pruning can make RMG keep searching and discovering important decomposition pathways.

PDD with eighteen carbons, is one of the largest systems RMG has modelled; compared with smaller molecules, it has much more reacting sites, higher number of species RMG has to keep track of and memory consumption. Consistent with previous testing cases, PDD job usually crashes around 200 species (hitting 8 Gb RAM limit) using RMG original algorithm (see Figure 10).

The mechanism misses some decomposition pathways, leading to slower conversion compared with

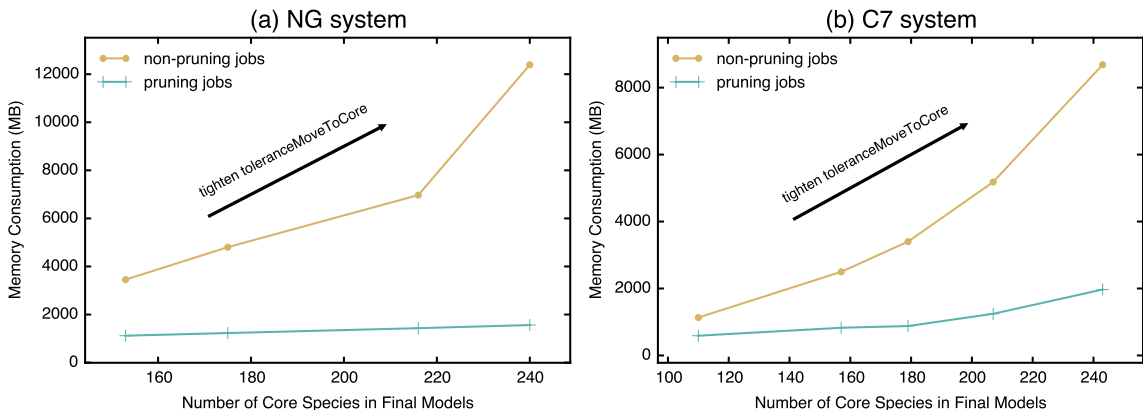


Figure 8: Comparison of RAM usage between original algorithm and pruning algorithm in two testing systems

Table 2: Modeling capacity is greatly extended by pruning algorithm for both natural gas and n-heptane systems

system	maximum species number in model (RAM requirement) with original algorithm	maximum species number in model (RAM requirement) with pruning algorithm
NG	220 (12 Gb)	$\geq 433$ (5 Gb)
C7	230 (8 Gb)	$\geq 531$ (4 Gb)

experiment observations (see Figure 12); one important missing pathway (see Figure 11) is PDD decomposes to pentadiene after more than five consecutive steps, which can actively react with styrene and create radicals through reverse disproportionation reactions.

Original RMG (no pruning) has to explore and store whole species space with radius of  $> 5$  steps to be able to find pentadiene but fails in the middle with memory filled up. With pruning turned on, RMG focuses on the most significant decomposition pathways, explores bigger species space and selects relevant pathways more efficiently, eventually discovers pentadiene pathway (Figure 11) with less than 3 Gb memory consumption, making the prediction of PDD conversion greatly improved (Figure 12).

The more detailed explanation of remaining discrepancy between PDD conversion experiment and prediction is out of scope of this paper; besides missing pathways, inaccurate thermochemistry, kinetics estimation in mechanism can be the remaining causes of discrepancy.

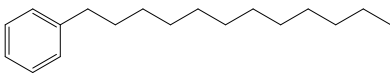


Figure 9: PDD (C18H30); phenyl dodecane

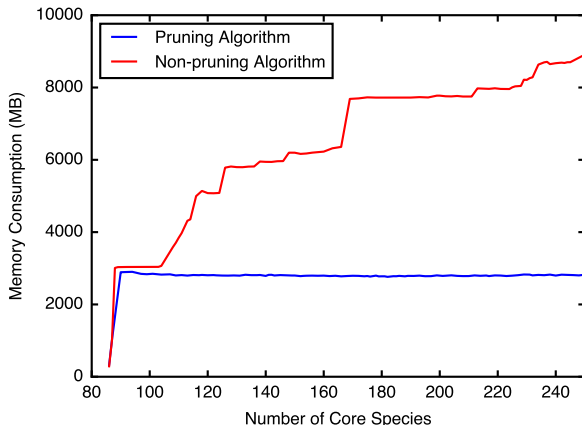


Figure 10: Pruning vs. Non-pruning memory consumption for phenyl dodecane decomposition

## 4. Discussion

### 4.1. Trade-off between effectiveness and accuracy

We are concerned about two aspects of the pruning algorithm: its **effectiveness** in reducing memory demands and its **accuracy** regarding final mechanisms generated. Quantitatively **effectiveness** and **accuracy** defined as follows:

$$\text{memory effectiveness} = \frac{RAM_{NP}}{RAM_P}$$

$$\text{accuracy} = \frac{|Species_P \cap Species_{NP}|}{|Species_{NP}|}$$

where  $RAM_{NP}$  and  $RAM_P$  are memory requirements for building a model using the non-pruning model and pruning algorithms respectively,  $Species_{NP}$  and  $Species_P$  are species sets in non-pruning model and pruning model respectively.

In the cases presented in Figure 8, the final models produced by RMG-Py are exactly the same whether or not pruning is used, i.e. they have maximal accuracy = 1. In other extreme cases, pruning reaches maximal memory effectiveness by deleting all the edge species, leading to a very different final model from the non-pruning case. In pruning algorithms, `toleranceKeepInEdge` is the key handle balancing these two aspects: a loose `toleranceKeepInEdge` reduces accuracy, but also reduces RAM requirement.

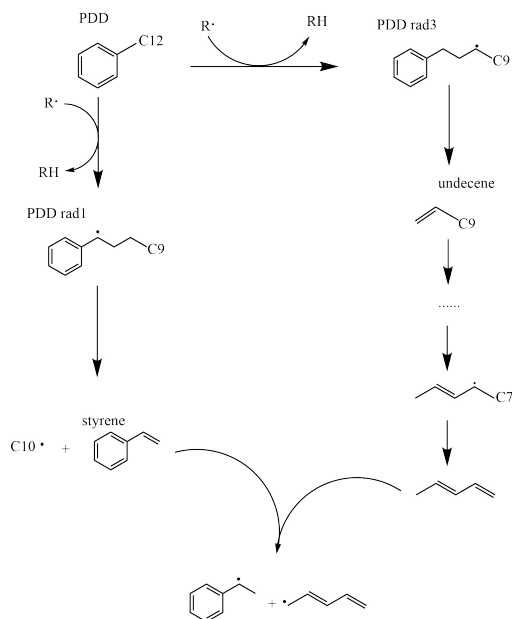


Figure 11: Important PDD decomposition acceleration pathway found by pruning

Scenario	Core species in final model	Core reactions in final model
non-pruning	153	5491
minCoreSizeForPrune=0	157	5840
minCoreSizeForPrune=50	153	5491

Table 3: Premature pruning impact. This example is NG system using `toleranceMoveToCore=0.3` and `toleranceKeepInEdge=0.01`

Thus determining the value of `toleranceKeepInEdge` is crucial for pruning performance. It turns out for complicated NG and C7 systems (with small `toleranceMoveToCore`) that `toleranceKeepInEdge` being smaller than 1/10 of `toleranceMoveToCore` usually gives good memory effectiveness and maintains the same models (accuracy=1), as Figure 13 and Figure 14 indicate.

By applying this rule of thumb (choose 1/10 of `toleranceMoveToCore` for `toleranceKeepInEdge`) to phenyl dodecane study, the pruning model not only keeps all the important pathways originally included in non-pruning model, but also discovers the pentadiene pathway that we discussed earlier.

Two additional components were designed to secure pruning **accuracy**. During early iterations, incomplete models usually result in inaccurate edge flux estimation, making the algorithm more likely to prune important species than it is when models are close to completeness (see Table. 3). It is observed that pruning at an early stage usually loses important species. Once an important species

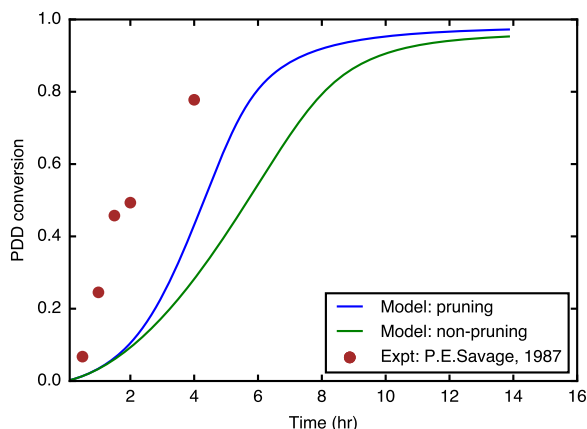


Figure 12: Pruning vs. Non-pruning phenyl dodecane conversion prediction

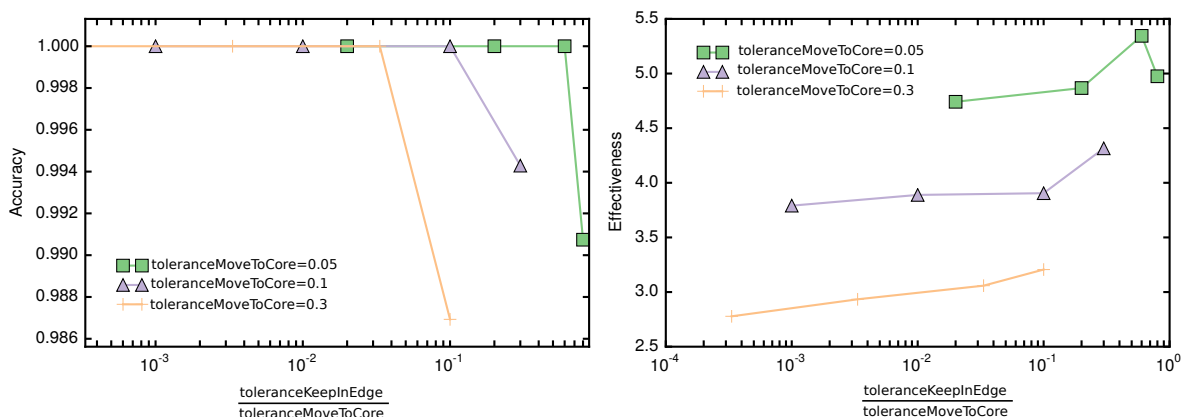


Figure 13: Effect of toleranceKeepInEdge in balancing pruning effectiveness and accuracy for NG system. Accuracy = 1 if pruning algorithm generates exactly identical model as non-pruning one does. Memory effectiveness is RAM reduction factor.

is lost, the model often grows in strange directions, including species which are not really important in the physical system. To prevent such undesired pruning, user-specified `minCoreSizeForPrune` is added to the pruning algorithm; no pruning is performed if the core species number is less than `minCoreSizeForPrune`. In both tested systems, `minCoreSizeForPrune = 50` is a good choice.

A second consideration is edge species eligibility to be pruned. For a newly generated edge species, its associated reaction network is usually not fully developed, and the consequent low flux usually misleads pruning. To avoid that, RMG records the age of each edge species (“age” = number of iterations since the species is first identified) so that only those with age larger than the user-specified `minExistIterationForPrune` are eligible to be pruned. We recommend setting `minExistIterationForPrune` to 3.

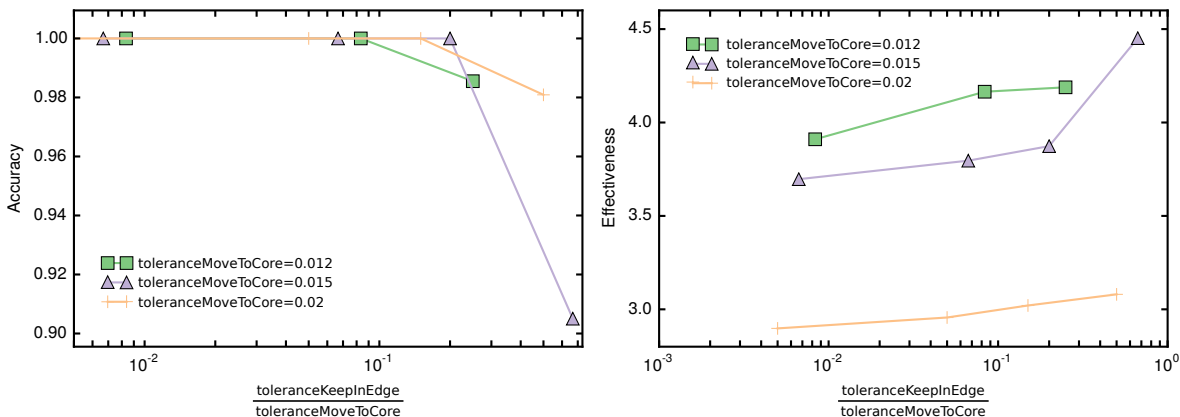


Figure 14: Effect of toleranceKeepInEdge in balancing pruning effectiveness and accuracy for C7 system. Accuracy = 1 if pruning algorithm generates exactly identical model as non-pruning one does. Memory effectiveness is RAM reduction factor.

#### 4.2. Flux evaluates species importance

Pruning algorithms rely on accurate evaluation of edge species' importance. From Figure 15, we can clearly see by ranking all the edge species using maximal normalized flux  $\max_t \left( \frac{\text{flux}_i(t)}{R_{\text{char}}(t)} \right)$ , that most species (colored in green) have small fluxes ( $\leq 10\%$  of toleranceMoveToCore). Our pruning algorithm removes them (dotted bars), based on the assumption that the fluxes to these minor byproducts are unlikely to change by orders of magnitude as the model is refined.

The original non-pruning algorithm [7] is designed conservatively, retaining all the edge species so if the flux towards any of the species gets large enough, that species will be added to the model. Our experience is that the reacting fluxes in the core and towards the edge species sometimes change significantly early in the mechanism-generation process, but after the major species have been included in the core, most of the major species concentrations and computed fluxes stabilize. It is unlikely that any of the fluxes will change by orders of magnitude because a few more minor species have been included in the model.

As shown in Figure 15, the fluxes towards most of the edge species are tiny, ten or more orders of magnitude smaller than the core species fluxes. It is therefore safe to delete these negligible species. Note that if a new reaction pathway towards a deleted edge species is discovered, the edge species will be resurrected, and maintained on the edge for at least minExistIterationForPrune to allow a fair re-assessment of its kinetic significance. We've observed that with reasonable tolerance values, the original rate-based algorithm and this pruned version yield exactly the same final models, but with very different memory requirements.

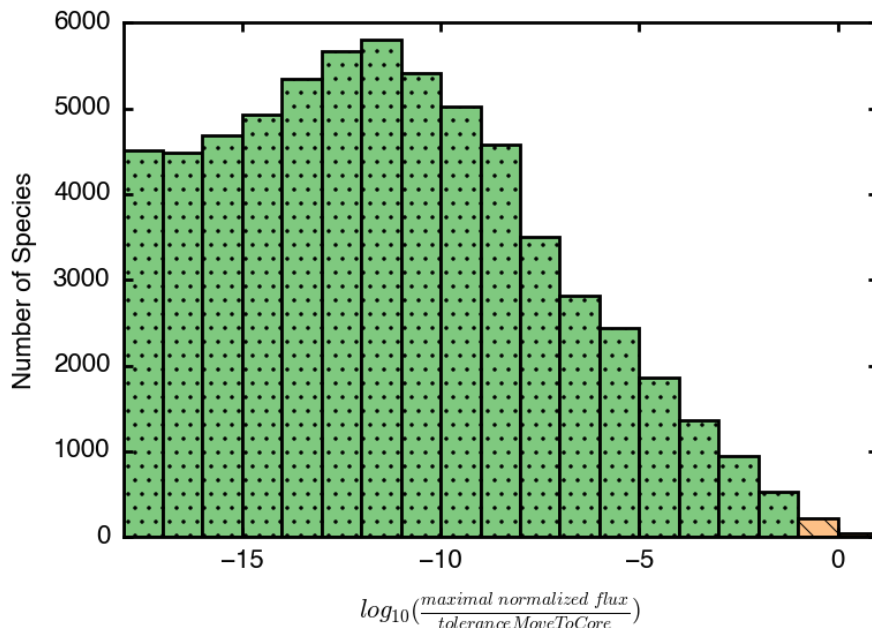


Figure 15: Maximal normalized flux ( $\max_t \left( \frac{\text{flux}_i(t)}{R_{char}(t)} \right)$ ) distribution of edge species at 180<sup>th</sup> iteration in non-pruning scenario of C7 system. With pruning most of the species on the left would be deleted, freeing a lot of memory

## 5. Conclusion

The memory limitation associated with RMG is largely mitigated by implementing a new algorithm introduced in this paper. By first selecting unimportant edge species and pruning them and their reactions, the memory consumption was reduced by a factor of 4 for cases where the original RMG algorithm runs into memory shortage. Several special considerations regarding pruning eligibility were applied to reduce the risk of mistaken pruning caused by model incompleteness. With the new pruning algorithm, it is practical to generate converged models for more complicated systems (high carbon number, high equivalence ratio, or low `toleranceMoveToCore`) than was possible in the past.

The new pruning algorithm performed well with two typical combustion systems having distinct chemical behaviors. It saves a significant amount of memory without affecting the final mechanism accuracy at all, since all the species pruned are negligible. In addition, pruning algorithm was able to discover missing pathways for a real application (phenyl dodecane thermal decomposition) that original RMG failed due to high memory consumption. Follow-up study of the trade-off between effectiveness and accuracy suggests an appropriate ranges of the tolerances for RMG-Py users to employ to execute the pruning algorithm properly.



## Acknowledgments

We gratefully acknowledge financial support from the US Department of Energy, Office of Basic Energy Sciences, Division of Chemical Sciences, Geosciences, and Biosciences under Contract DE-FG02-98ER14914.

## References

- [1] E. Ranzi, T. Faravelli, P. Gaffuri, A. Sogaro, Low-temperature combustion: Automatic generation of primary oxidation reactions and lumping procedures, *Combustion and Flame* 102 (12) (1995) 179–192. doi:10.1016/0010-2180(94)00253-0.
- [2] F. Battin-Leclerc, Development of kinetic models for the formation and degradation of unsaturated hydrocarbons at high temperature, *Physical Chemistry Chemical Physics* 4 (11) (2002) 2072–2078. doi:10.1039/B110563A.
- [3] L. J. Broadbelt, S. M. Stark, M. T. Klein, Computer Generated Pyrolysis Modeling: On-the-Fly Generation of Species, Reactions, and Rates, *Industrial & Engineering Chemistry Research* 33 (4) (1994) 790–799. doi:10.1021/ie00028a003.
- [4] J. Song, Building robust chemical reaction mechanisms : next generation of automatic model construction software, Thesis, Massachusetts Institute of Technology, thesis (Ph. D.)—Massachusetts Institute of Technology, Dept. of Chemical Engineering, 2004. (2004).
- [5] C. W. Gao, J. W. Allen, W. H. Green, R. H. West, Reaction Mechanism Generator: Automatic construction of chemical kinetic mechanisms, *Computer Physics Communications* 203 (2016) 212–225. doi:10.1016/j.cpc.2016.02.013.
- [6] D. J. Klink, L. J. Broadbelt, Mechanism reduction during computer generation of compact reaction models, *AIChE Journal* 43 (7) (1997) 1828–1837. doi:10.1002/aic.690430718.
- [7] R. G. Susnow, A. M. Dean, W. H. Green, P. Peczak, L. J. Broadbelt, Rate-Based Construction of Kinetic Models for Complex Systems, *The Journal of Physical Chemistry A* 101 (20) (1997) 3731–3740. doi:10.1021/jp9637690.