A Multi-Level Method for the Steady State Solution of Markov Chains

Claudia Isensee, Graham Horton[*]
Otto-von-Guericke Universität Magdeburg

Abstract

This paper illustrates the current state of development of an algorithm for the steady state solution of continuous-time Markov chains. The so-called multi-level algorithm utilizes ideas from algebraic multigrid to provide an efficient alternative to the currently used Gauss-Seidel and successive overrelaxation methods. The multi-level method has been improved through several iterations, so that it is now able to solve several classes of Markov chains robustly and efficiently. Among these are Markov chains with heterogeneous transition rates and ones with almost identical transition rates. Experiments were used to verify the improvements throughout the iterations and the advantages in comparison to the usual methods.

1 Introduction

Markov chains are one of the most important kinds of models in Simulation. The steady state solution of continuous-time Markov chains (CTMC) is of interest in this paper. Gauss-Seidel and SOR are the most widely used methods for that purpose, but they may need a considerable number of iterations to reach an accurate solution. Not only does the computation time per iteration increase with growing problem size, but also the number of iterations needed may depend on the size of the problem. Therefore a new robust algorithm is needed for the steady state solution of continuous-time Markov chains. An introduction to the topic of Markov chains and solution methods can be found in [Ste94].

This paper describes a method for the steady state solution of CTMCs that has been developed, as well as the additions that have been made to the method so far.

2 Motivation

For two specific classes of Markov chains, SOR and Gauss-Seidel may perform particularly badly. These include CTMCs with heterogeneous transition rates and CTMCs with almost identical transition rates. An example for the former class can be found in Figure 1 left. This is a so called 'nearly completely decomposable' (NCD) Markov chain, and it consists of groups of nodes that are strongly connected among each other and have weak connections between the groups. Even for this small example, the number of necessary iterations and hence the time to convergence of the chain explodes as ϵ tends towards zero.

[*]Fakultät für Informatik, Institut für Simulation and Graphik, D-39016 Magdeburg, Germany

This happens, because the first and second two nodes are connected strongly, and probability can flow freely between them. Therefore Gauss-Seidel can smooth their relative probabilities within a few steps. But the connection between the second and third node is weak in comparison to the other edges, and it takes many more iterations to transfer a significant amount of probability between them, because the nodes influence each other only marginally, and probability flows slowly between them.

Figure 1 right shows a Markov chain with identical transition rates which results from a M/M/1 queuing system with identical arrival and service rates. As the length of the Markov chain grows, the number of iterations needed increases sharply. This is due to the fact that in principle the probability has to be distributed evenly throughout the chain, but it takes a number of iterations to propagate it through the whole chain.

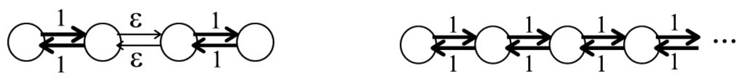

Figure 1: Nearly completely decomposable Markov chain (left), one-dimensional Markov chain with identical transition rates (right)

The goal was to develop a robust general-purpose method for the steady state solution of CTMCs. The first version of the current algorithm is the ’multi-level aggregation’ (ML-A) method, that was developed by Horton and Leutenegger and introduced in [HL94]. It is based on ideas from algebraic multigrid, where a good introduction can be found in [RS87]. The ML-A method was developed to utilize the structure of Markov chains with heterogeneous transition rates. Especially for NCD chains, computation time decreased radically, but for Markov chains with identical transition rates the improvements were minimal. Nevertheless, the multi-level approach was promising and highly adaptive, and so it was used as a starting point for a general-purpose algorithm.

3 The Multi-Level Algorithm

The general strategy of the multi-level algorithm is to create a series of ever smaller Markov chains from the original one. By working on all of these simultaneously within each iteration, the algorithm is much faster than conventional ones. This works, because the Multi-Level algorithm combines nodes that influence each other strongly and whose values can be smoothed by a few Gauss-Seidel steps. They are mapped to a common node on the so-called coarser level, thereby removing the disparity between strong and weak connections. Probability can flow freely between the nodes on the coarse level, therefore an equilibrium can be reached in fewer iterations. By applying this process recursively, the chain is coarsened, and any disparities between transition rates are compensated. By creating a series of successively smaller coarse chains, one-dimensional Markov chains with identical transition rates can also be solved.

However, it is of vital importance how the nodes are aggregated here, which ones are combined, or even split between their neighbors.The process of calculating that mapping is called aggregation. It is an important part of the algorithm, and will be described in Section 3.2.

The original steady state solution problem of a Markov chain can be specified as follows:

$$Q\pi = \vec{0} \tag{1}$$

where π is a probability vector, and $Q = [q_{ij}]_{n\times n}$ is the generator matrix of the CTMC. Gauss-Seidel and SOR solve this linear system of equations by subsequent iterations through the whole range of unknowns. To reduce the number of unknowns in order to obtain a smaller Markov chain, the chain has to be coarsened. The nodes on the fine level are also called children, as the ones on the coarser level are parents.

The *restriction* maps the generator matrix Q, as well as the probability vector π, from the fine to the coarse level, and can be visualized as matrix R. The row sums of the matrix are one, which ensures, that the probability of a fine node is always transferred completely to the coarse level, and the result is still a probability vector. A child node can, however, have multiple parents.

The *prolongation* maps the probability vector π from the coarse to the fine level and can be expressed as matrix P. A parent node can have multiple children, and again the probability has to be transferred from the coarse to the fine level completely, which results in column sums of one.

$$R = [r_{ij}]_{n_{(l)}\times n_{(l-1)}} \in \mathbf{R}^{n_{(l)},n_{(l-1)}} \tag{2}$$

$$P = [p_{ij}]_{n_{(l-1)}\times n_{(l)}} \in \mathbf{R}^{n_{(l-1)},n_{(l)}} \tag{3}$$

To iterate through the Markov chain on each level Gauss-Seidel is employed, which computes the new probability of a node as described in [Ste94] (page 128)

Some Gauss-Seidel steps are performed on every level before restriction and after prolongation to smooth the error. On the lowest level, with the smallest chain, Gauss-Seidel is used to solve the problem exactly.

The complete algorithm is structured as follows. The first step is to compute the mappings of the original Markov chain to the coarser one according to the aggregation strategy, as well as all subsequent mappings until a minimal chain with only two or three nodes is obtained. A few Gauss-Seidel steps are performed on every level before calculating the mapping. Then the iteration starts on the highest level and propagates recursively through the coarser ones according to the following scheme.

The first step on every level is to test whether the lowest level has been reached, and if yes, to solve the chain using Gauss-Seidel. Otherwise a few Gauss-Seidel steps are performed to smooth the probability vector. Afterwards Q and π for the next coarser level are computed via restriction. The steady state vector of the Markov chain at the next level is then processed in a recursive fashion. The new coarse probability vector is used to correct the values of the fine one via prolongation. As a last step, the vector is normalized, and some more Gauss-Seidel steps are performed for smoothing. Schematically the recursive structure of an iteration step can be represented as follows:

```
ML_iteration (level)
begin
  if (level = 0)
    solve (level);
  else
    smoothing    (level);
    restriction  (level, level-1);
    ML_Iteration (level-1);
    prolongation (level-1, level);
    correction   (level);
    smoothing    (level);
end
```

It has not been mentioned yet, according to what rules the mapping from one level to the next is computed. This is a central part of the algorithm, and can contribute significantly to its success. Nodes could be aggregated randomly or the structure of the chain could be exploited to create meaningful groupings. Different approaches and strategies will be introduced in Section 3.2.

3.1 Restriction and Prolongation

Restriction and prolongation are the central operators when mapping the values (Q, π) of a fine chain to a coarse one or the other way around. A simple example for such a mapping is shown in Figure 2 left, where (l) and $(l-1)$ denote the fine and coarse levels, respectively.

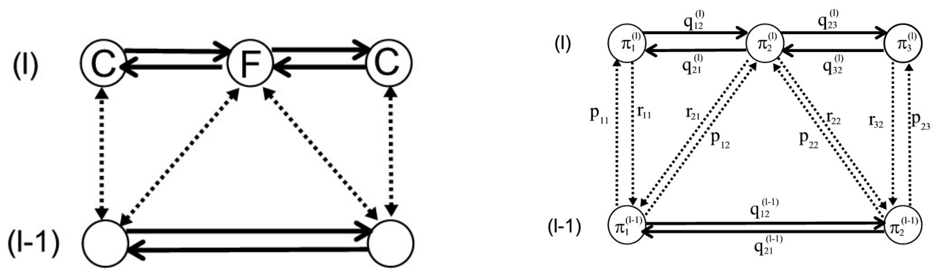

Figure 2: Restriction of a simple Markov chain

The two operators can also be expressed as matrices R and P, and have the following properties. A fine node $i^{(l)}$, that only has one parent $j^{(l-1)}$, will be mapped to the next level as a whole and is called a C-node. The matrix elements are as follows for these nodes:

$$r_{ij} = 1 \; and \; r_{ik} = 0, \forall k \neq j \tag{4}$$

$$p_{ji} = 1 \; and \; p_{ki} = 0, \forall k \neq j \tag{5}$$

A node $i^{(l)}$ that has several parent nodes $j^{(l-1)}$, and whose probability will be split to the parents of its neighbors $k^{(l)}$, is called an F-node. The matrix elements for such nodes are computed as follows, where N_i is the set of neighbors, whose parents $i^{(l)}$ is mapped to, and $j^{(l-1)}$ is the parent node of neighbor $k^{(l)}$:

$$r_{ij} = \frac{q_{ik}}{\sum_{m \in N_i} q_{im}} \qquad\qquad p_{ji} = \frac{q_{ki}}{\sum_{m \in N_i} q_{mi}} \tag{6}$$

Through restriction, the generator matrix Q, as well as the probability vector π, are mapped to the coarse level. The new elements are computed as follows:

$$\pi_j^{(l-1)} = \sum_{i=1}^{n_{(l)}} r_{ij} \pi_i^{(l)} \tag{7}$$

$$q_{ij}^{(l-1)} = \frac{\sum_{k=1}^{n} r_{ki} \pi_k^{(l)} (\sum_{m=1}^{n} q_{km}^{(l)} r_{mj})}{\sum_{k=1}^{n} r_{ki} \pi_k^{(l)}} \tag{8}$$

Prolongation just propagates the corrections "upwards" to the fine level. The corrected probabilities $\pi_i'^{(l)}$ of node i are a product of the original probability $\pi_i^{(l)}$ of the node and a correction factor, which is the sum of the probability changes of the parent nodes of i:

$$\pi_i'^{(l)} = \pi_i^{(l)} \sum_{j=1}^{n_{(l-1)}} p_{ji} \frac{\pi_j'^{(l-1)}}{\pi_j^{(l-1)}} \tag{9}$$

The labeling during restriction and prolongation is again clarified in Figure 2 right.

Restriction and prolongation are important when computing the new values for the nodes on the coarse level, and when propagating the corrections to the fine level, but the calculation of the mapping itself is much more crucial. How the decision is made, whether to split a node to its neighbors or not, or with which of its neighbors it should be aggregated is part of the so called aggregation strategy. Several different ones will be described in the next section.

3.2 Aggregation Strategies

The aggregation method is the heuristic part of the multi-level algorithm, and has a major influence on its performance. The decisions which nodes are going to be aggregated and which might be split to their neighbors are made according to the aggregation strategy. The main goal of the aggregation strategy is to aggregate the nodes in a way, that the previously described Gauss-Seidel property is taken advantage of, that nodes whose relative errors can be smoothed out in a few iterations are combined and mapped as one to the next level, and that nodes who are equally strongly connected to more than one neighbor might be split to them.

This is also the part of the algorithm that has undergone the most changes since the original version was published in [HL94]. In that paper, two aggregation strategies are described. One is very simple, it always aggregates pairs of neighboring states, according to their numbers. The other strategy is very specific and adapted to Markov chains generated from a tandem queueing system. The authors mention that it would be useful to exploit the structure of certain chains if it is known, or to aggregate by strong connections, if no prior knowledge of the chains structure is available. This original version of the algorithm only includes the concept of C-nodes, which are mapped to the coarse level as a whole.

In [LH95], the ML-A algorithm is applied to NCD Markov chains, and the chosen strategy was to aggregate by strong connections. It was also defined that a coarse node should have at most three children, so that the number of nodes does not decrease too fast, otherwise the advantage of having intermediate levels would be lost.

The strategy of aggregation by strong connections works very well for Markov chains with heterogeneous transition rates, such as NCD Markov chains. The performance of the algorithm proved to be much better than that of conventional methods, even iteration rates that are independent of problem parameters could be achieved, which is the optimal result. Markov chains with almost identical transition rates on the other hand only showed a moderate improvement in computation effort when solved with the multi-level algorithm.

In [Lab97], an extension of the algorithm was proposed (ExtML). Nodes which are connected equally strongly with more than one neighbor may also be split among these. This necessarily results in a change of the restriction and prolongation formulae. The strategy to determine whether a node is connected equally strong to some neighbors is solely based on thresholds and local transition rates. Specific patterns of the Markov chain are not utilized, and so the strategy is still not perfect. Nevertheless, for one-dimensional Markov chains with identical transition rates (Figure 1 right), the performance could be improved drastically. A strategy that alternates C-nodes and F-nodes is found by the algorithm. It represents the optimal solution, since some coarsening has to take place, but no intelligent groupings can be made. A node is always connected equally strong to both of its neighbors. Therefore every second node is split equally to its neighbors, and the problem of choosing one aggregation partner is resolved.

Other patterns, like the one in Figure 3 are not detected by the ExtML algorithm, even though the optimal strategy is again to alternate C- and F-nodes. This grid consists of one-dimensional, parallel strands which are strongly connected within, but have only a weak coupling to the other strands. To aggregate nodes from different strands would be false, since they only influence each other weakly in comparison to neighboring nodes from the same strand. The first logical step is to aggregate all nodes from one strand, and since these have the same structure, as the one-dimensional chains mentioned above, the strategy of alternating C- and F-nodes should also be employed here. Once the parallel strands have been collapsed, only one vertical strand remains. The disparity between weak and strong couplings is now removed, and the strand can also be collapsed, again employing the same strategy as above.

Therefore, a new aggregation strategy was developed in [Ise03] (NewML), which should detect and exploit patterns such as the one in Figure 3 left. The general idea of the method is to first decompose the chain into groups of strongly connected nodes, and then try to find the optimum strategy for the coarsening of the individual groups. Thereby the advantages of the ML-A an NewML Algorithm are combined, heterogenous transition rates, as well as nearly identical transition rates are treated appropriately.

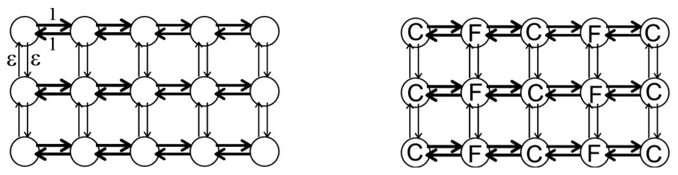

Figure 3: Markov chain with simple grid structure and known optimal coarsening

The first step of this strategy is to separate the edges of the Markov chain into sets which are deemed to be relevant or irrelevant for the further aggregation. Throughout the whole process of aggregation only these edges are considered, but they are not weighted against each other.

To further describe the algorithm, it is first necessary to specify a relevant edge. This definition is derived from one specific property of the Gauss-Seidel method, since that is still being used to compute the solutions on the coarsest level, and to smooth the probability vector π by a couple of iterations on each level. One Gauss-Seidel iteration consists of calculating the new probabilities of all nodes in the chain in a specific order. Newly calculated probabilities already affect the probabilities computed later during the same iteration. Consequently it is important in what order the nodes are processed.

If the transition rate, and therefore the flow of probability, is strong from node i to node j, and node i comes before j during the calculation, then any change in i is propagated to j during the same iteration, and the probability ratio of the nodes stabilizes within a few Gauss-Seidel steps. These nodes can be aggregated and treated as one on subsequent levels, since any change will only affect their relation to the other nodes surrounding them, but not between themselves. These nodes are strongly connected, and the edge from node i to node j is thus relevant. In other words, an edge that is strong in comparison to the other edges of its start and destination node, and that runs in the same direction as the order that Gauss-Seidel processes the nodes in, is relevant. This is emphasized in Figure 4, where the numbers of the nodes represent the Gauss-Seidel processing order.

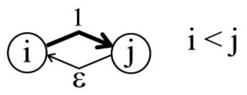

Figure 4: Definition of a relevant edge

After all edges have been categorized, this labeling is used to initially classify the nodes as either C or F, according to the numbers of their relevant neighbors. Only nodes that have more than one relevant neighbor can later be F-nodes, otherwise there is nothing to split them to. Thus, nodes with two or more neighbors are marked as F-nodes, all others become C-nodes. Ideally the chain will decompose into groups of nodes that are strongly connected among each other, but only weakly coupled with nodes from other groups. The current restriction formula can not deal with neighboring F-nodes that are connected by a relevant edge, therefore the number of F-nodes has to be reduced. A greedy strategy passes through the groups of nodes recursively, and marks all relevant neighbors of an F-node with C.

The now fully classified nodes can then be mapped to the coarse level. First all C-nodes are aggregated or mapped to the coarse level by themselves, and then all F-nodes are split to their C-neighbors' parents. Again the upper bound for C-children of a coarse level node is set to three. This is the only step that has to take irrelevant edges into account, specifically those edges parallel to relevant edges between F- and C-nodes.

An example of the described strategy is depicted in Figure 5. In the picture of the original chain on the left, the relevant edges have been emphasized. The picture on the right side shows the different groups, that the chain decomposes into, and the resulting coarse-level chain, as well as the mapping from the fine to the coarse level.

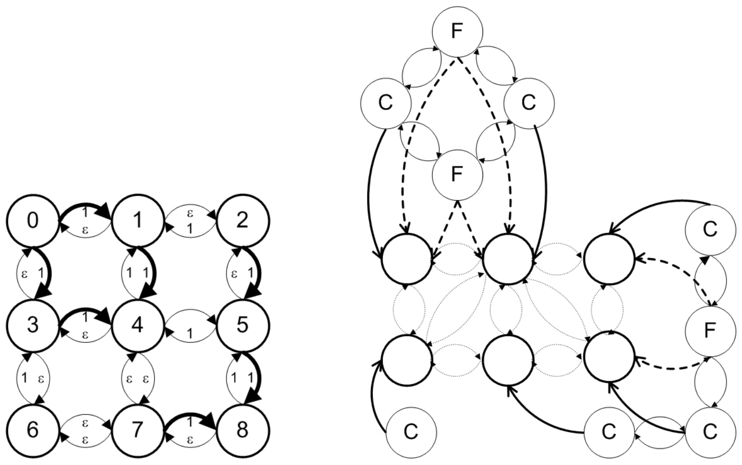

Figure 5: NewML aggregation strategy applied to example Markov chain

The strategy of the algorithm that takes the direction of an edge into account can lead to few or even no relevant edges after a couple of iterations. For aggregation, however, the relevant edges are essential, and any further coarsening would not be possible, or the number of nodes would only decrease minimally. The reason for this is, that there are still strong edges, but they run in the wrong direction. Hence, it is necessary to sort the Markov chain, in order to rotate the strong edges and make them relevant again, and thereby making the main flow of probability run in the order of Gauss-Seidel processing. A sorting algorithm that does exactly that is described in [Ise03] and has been integrated into the multi-level method. Another advantage of sorting the Markov chain by relevant edges is that the effectiveness of the Gauss-Seidel method increases, since changes are propagated along the strong edges during each iteration.

Schematically the algorithm looks like this:

```
1. Mark relevant edges of the  Markov chain.
2. If too many edges are irrelevant,
   sort chain, continue at step 1.
3. Mark nodes with less than two neighbors as C-nodes,
   mark all others as F-nodes.
4. Mark relevant neighbors of F-nodes as C-nodes.
5. Aggregate C-nodes and map them on their parent nodes.
6. Split F-nodes to their relevant neighbors.
   Pay special attention to parallel irrelevant edges.
```

The main idea of the multi-level algorithm is to aggregate nodes whose relative values can be smoothed by a few Gauss-Seidel steps, and then treat them as one node on subsequent levels. In addition to that, nodes that are strongly connected to more than one neighbor are divided up among these, and thereby a very good convergence rate can be achieved. The increments of the algorithm that were done, always included the advantages of the former versions. Thereby the class of Markov chains that can be solved efficiently with the multi-level method has increased step by step. In the next section experimental evidence is given for the advantages of the multi-level method.

4 Experiments

To test the properties of the different multi-level strategies and how they perform in comparison to a conventional algorithm, several experiments were conducted. Some conclusive results are presented in this section. The implemented algorithms are:

- the original multi-level method from [HL94], that uses aggregation only,
- the extended multi-level algorithm from [Lab97], that also uses splitting of nodes (ExtML),
- the multi-level method from [Ise03] with the new aggregation strategy (NewML),
- and the successive overrelaxation method, a derivative of Gauss-Seidel (SOR).

Each experiment tests some or all of the methods on a specific class of Markov chains. The test classes are:

- one-dimensional Markov chains with identical transition rates,
- two-dimensional grids similar to the one in Figure 3,
- and Markov chains with heterogeneous transition rates.

The performance measure chiefly used, is the number of floating point operations needed to achieve a decrease in the error of a factor of 1e-06. In addition to that the number of iterations needed to reach the same goal is measured.

4.1 A Markov Chain with Almost Identical Transition Rates

Figure 6 left shows that on one-dimensional Markov chains with identical transition rates, all multi-level methods perform better than SOR. The ExtML and NewML methods are faster than the ML-A algorithm (Figure 6 right).

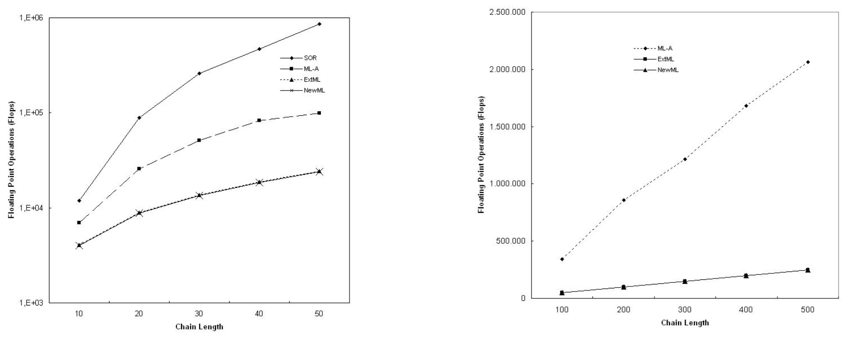

Figure 6: Number of floating point operations for one-dimensional Markov chains of different lengths with different solution methods, logarithmic and normal scaling

For two-dimensional grids with strong transition rates in one direction, and weak ones in the other, the ML-A and the ExtML method behave similarly. The NewML algorithm shows considerable performance improvements compared to the other two multi-level methods. Figure 7 left depicts the grids growing in the direction of the strong connections, and Figure 7 right shows the same results for grids growing in the direction of the weak connections.

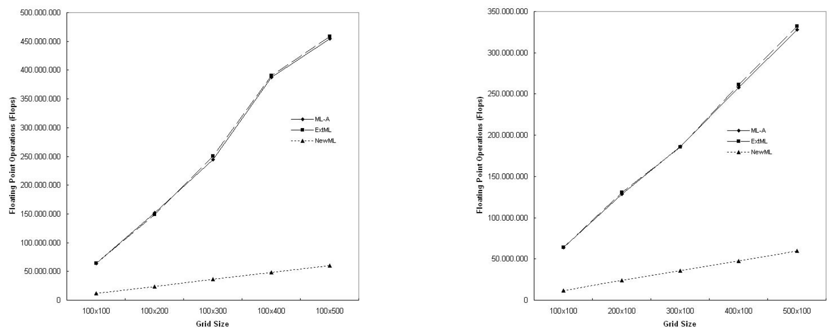

Figure 7: Number of floating point operations for grid-structured Markov chains of different size with different solution algorithms

The differences among the multi-level methods are caused by the different aggregation strategies. When using the optimal aggregation strategy for a specific structure, the number of iterations needed becomes independent of the size of the problem. For one-dimensional chains this goal is achieved already by the ExtML algorithm (Figure 8 left), but for grids only the NewML algorithm finds the optimal aggregation (Figure 8 left). In both cases the number of iterations needed to solve the problem when using the optimal aggregation strategy is four.

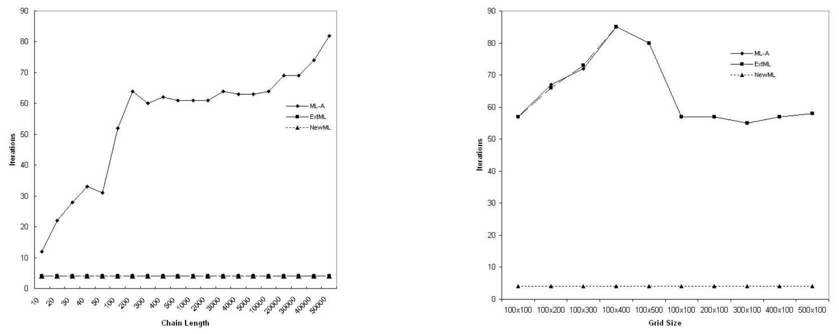

Figure 8: Number of iterations needed for different sized regularly structured Markov chains and different solution algorithms

4.2 A Markov Chain with Heterogeneous Transition Rates

For this experiment Markov chains were generated from a queuing network model of an interactive computer system, described in [Ste94] (pages 88/89). This is a model of a computer system with a variable number of terminals attached to it, that can only have one job each within the system. The test chains were generated by varying the number of terminals.

Figure 9 left shows, that all multi-level methods perform much better on this set of chains, than the SOR algorithm. There are no significant differences between the results of the multi-level methods, since the ML-A method was already designed to perform well on heterogeneous Markov chains (Figure 9 right).

5 Conclusion

The experiments showed that multi-level methods are much faster than the widely used SOR method on the tested groups of Markov chains. The ML-A method performs well on Markov chains with heterogeneous transition rates. The ExtML method extends this good performance to Markov chains with nearly identical transition rates. The NewML algorithm adds Markov chains with patterns with almost identical transition rates to the class of efficiently solvable Markov chains.

In addition, a sorting algorithm for Markov chains was developed, that not only improves the performance of the multi-level component of the algorithm, but that also enhances the

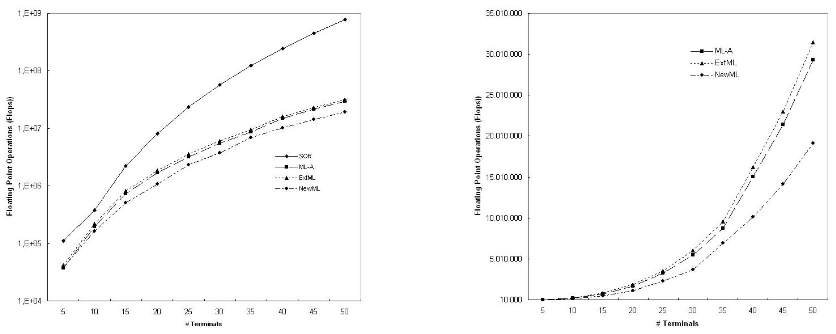

Figure 9: Number of floating point operations for different sized Markov chains with heterogeneous transition rates, logarithmic and normal scaling

convergence of the Gauss-Seidel method.

The multi-level method captures an ever-increasing number of Markov chain classes, and through further extensions and tuning as well as thorough testing, we hope to be able to extend it to a new general-purpose algorithm for the steady state solution of Markov chains.

References

[HL94] Graham Horton and Scott T. Leutenegger. *A Multi-Level Solution Algorithm for Steady-State Markov Chains*. In: ACM SIGMETRICS, 1994.

[Ise03] Claudia Isensee. *Aggregationsstrategien für ein Multilevel-Verfahren zur Lösung von Markov-Ketten*. Thesis, Otto-von-Guericke-Universität Magdeburg, 2003.

[Lab97] Ulf Labsik. *Algorithmische Erweiterung des Multi-Level-Verfahrens zum Lösen von Markov-Ketten*. Thesis, Friedrich-Alexander-Universität Erlangen-Nürnberg, 1997.

[LH95] Scott T. Leutenegger and Graham Horton. *On the Utility of the Multi-Level Algorithm for the Solution of Nearly Completely Decomposable Markov Chains*. In: Second Int Workshop on the Numerical Solution of Markov Chains, 1995.

[RS87] J. W. Ruge and K. Stüben. *Algebraic Multigrid*. In: Stephen F. McCormick, Ed., Multigrid Methods. SIAM, Philadelphia, PA, 1987.

[Ste94] W. J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, Princeton, NJ, 1994.