M3A: Marked MAP Matching Algorithms

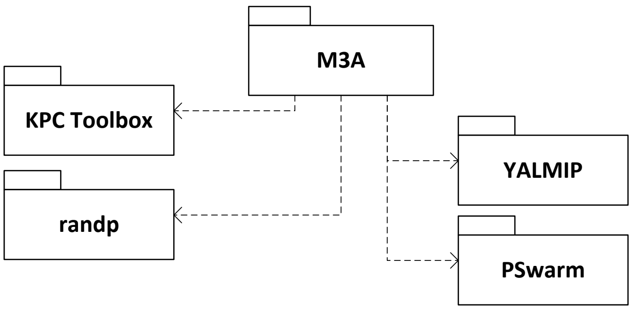
Andrea Sansottera, Giuliano Casale, Paolo Cremonesi

Markovian Arrival Processes (MAPs) are a class of stochastic processes used to model the arrivals at a queuing system. Marked MAPs (MMAPs) are an extension of MAPs that allow to model arrivals of multiple customer classes. The usefulness of MMAPs is due to the existence of efficient techniques for the analytical solution of queuing system whose arrival process is a MMAP.

The M3A library is a set of Matlab functions designed for computing the statistical descriptors of MMAPs and fitting marked traces with MMAPs. The latter problem is non-trivial due to the typically complex and nonlinear relationships between the statistical descriptors of the marked trace and the parameter of the MMAP. If you use M3A for your research, please cite the following paper:

- *Fitting second-order acyclic Marked Markovian Arrival Processes, Sansottera A., Casale G. and Cremonesi P., Dependable Systems and Network, 2013* (link)

Every MMAP has an underlying MAP, obtained by neglecting the labels of arrivals. M3A requires the KPC Toolbox for the computation of the characteristics of MAP underlying an MMAP. M3A also relies on other third party software: randp, YALMIP and PSwarm. YALMIP is used for the non-convex optimization problems involved in some of our fitting algorithms. PSwarm is optionally used for fitting the underlying unmarked proess with a second-order MAP.



The mathematical representation of a MAP of order N is composed of two NxN matrices: a sub-stochastic matrix D_0 and a non-negative matrix D_1 such that $D_0 + D_1$ is stochastic. Similarly, a MMAP of order N with M classes is represented as a sub-stochastic matrix D_0 and M non-negative matrices D_{11},D_{12},...,D_{1M} such that $D_0 + D_{11} + D_{12} + ... + D_{1M}$ is stochastic.

In Matlab, we represent a MAP as a cell array of size 1x2, whose elements are D_0 and D_1. Similarly, we represent an MMAP with M classes as a cell a array of size 1x(2+M), where the first two elements are D_0 and $D_1 = D_{11} + D_{12} + D_{1M}$ and the other elements are the D_{1c} matrices. Observe that, adopting this representation, every function designed to operate on MAPs also works on MMAPs, since the first two elements of the cell array representing an MMAP constitute a valid MAP (the MAP *underlying* the MMAP). We represent a trace as a column vector T of the inter-arrival times. A marked trace is also characterized by a second column vector C, of the same length, with the class labels associated with each arrival.

To compute the theoretical statistical descriptors of an MMAP and the empirical statistical descriptors of a marked trace, M3A implements several functions. The functions for the computation of the descriptors of MMAPs operate on both symbolic and numeric MMAPs. The implemented functions are as follows:

M = mmap_forward_moment(MMAP, orders)	M = mtrace_forward_moment(T,C, orders)
M = mmap_backward_moment(MMAP, orders)	M = mtrace_backward_moment(T,C, orders)
M = mmap_cross_moment(MMAP, order)	M = mtrace_cross_moment(T,C, order)
P = mmap_pc(MMAP)	P = mtrace_pc(T,C)
PP = mmap_sigma(MMAP)	PP = mtrace_sigma(T,C)
PPP = mmap_sigma2(MMAP)	PPP = mtrace_sigma2(T,C)
GAMMA = map_gamma(MAP)	GAMMA = trace_gamma(T)

The first three pairs of functions computed forward, backward and cross moments, which are conditional moments defined on the basis of the class labels. See the above mentioned paper for the definitions. The *[mmap|mtrace]_pc* functions compute the arrival probabilities of each class. The *[mmap|mtrace]_sigma* functions compute a matrix containing the one-step class transition probabilities, e.g., the probability of an arrival of class 1 followed by an arrival of class 2. The *[mmap|mtrace]_sigma2* functions compute a 3D matrix containing the two steps class transition probabilities. Finally, the *[map|trace]_gamma* functions, designed to operate on MAPs and unmarked traces, compute an estimate of the auto-correlation decay rate (assuming a geometric autocorrelation function, which is exact for a second-order MAP). Other functions for the computation of characteristics of the underlying MAP are provided by the KPC Toolbox.

For MMAPS, the following functions are also available:

[T,C] = mmap_sample(MMAP, samples)
TF = mmap_isfeasible(MMAP)
pie = mmap_pie(MMAP)
TF = mmap_issym(MMAP)

The first one samples a trace from a given MMAP. The second one checks whether a MMAP representation is valid. The third one computes the interval-stationary state probabilities of the MAP conditioned on the class of the preceding arrival. The last one checks whether a given MMAP is a symbolic or numeric object.

As of now, M3A implements the following method to fit a marked trace:

1. fitting a second-order MAPH (a special case of the MMAP lacking correlation in the underlying process) based on the class probabilities and the first-order backward moments (any number M of classes, function *maph2m_fit_ trace*);

2. fitting a second-order MMAP based on the class probabilities, first-order forward moments and backward moments (any number M of classes, function *mamap2m_fit_gamma_bf_trace*);
3. fitting a second-order MMAP based on the class probabilities, first-order forward moments and one -step class transition probabilities (only two classes, function *mamap22_fit_gamma_fs_trace*);
4. fitting a second-order MMAP based on the class probabilities, first-order backward moments and one -step class transition probabilities (only two classes, function *mamap22_fit_gamma_bs_trace*);
5. fitting a MMAP of order MMAP $3M^2$, matching the forward and backward moments up to order 3, the class probabilities, one-step class transition probabilities and two-step class transition probabilities (any number M of classes, function *mmap2m_mixture_fit_trace*).

Finally, we provide the function *mamap2m_fit_trace* that automatically selects one of methods 1,2,3,4 depending on the trace characteristics and a vector of user provided weights that express the preference for forward moments, backward moments and class transition probabilities. Method 5 is described in a journal paper that is currently under peer review.

Lower level functions and functions to fit the underlying processes are represented in the next page. Notice that some of the fitting functions, whose name ends with *_*mmap*, can be used to fit the theoretical descriptors of an MMAP instead of the empirical descriptors of a trace.

Examples

Define an MMAP and then checks whether it is feasible or not.

```
>> MAP = {[-1,1/3;0,-2],[2/3 0; 1/4 7/4]};
>> MMAP = {MAP{1},MAP{2},MAP{2}.* [2/3 0; 0 2/7],MAP{2}.*[1/3 0; 1 5/7]};
>> mmap_isfeasible(MMAP)
ans = 1
```

Fit the theoretical class probabilities, forward and backward moments of the MMAP.

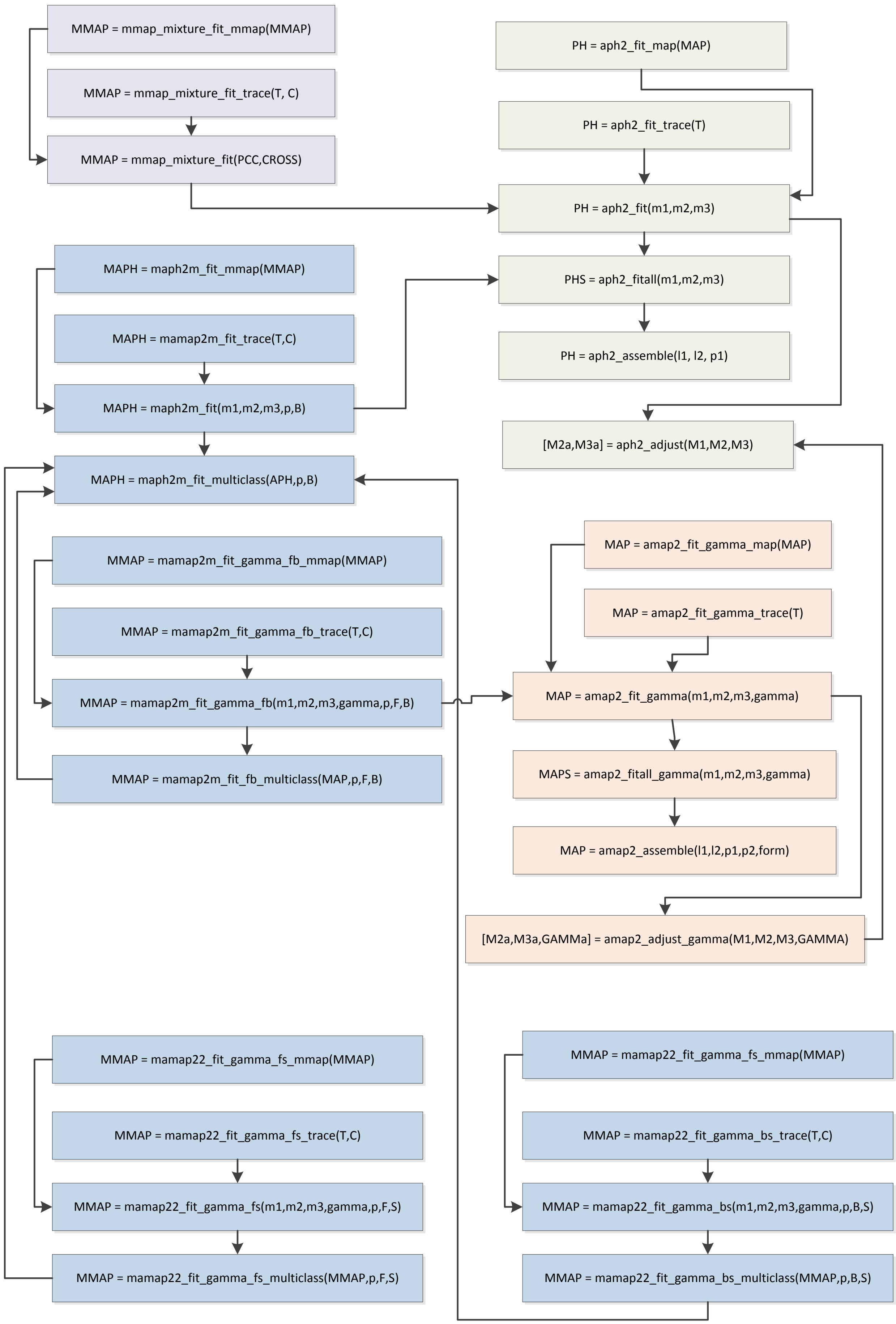
```
>> FIT = mamap2m_fit_gamma_fb_mmap(MMAP)
FIT =
    [2x2 double]    [2x2 double]    [2x2 double]    [2x2 double]
```

Sample a trace from the original MMAP, then fit the empirical forward moments, class probabilities and class transition probabilities. Finally, it prints the difference between the forward moments of the trace and the forward moments of the fitted MMAP.

```
>> [T,C] = mmap_sample(MMAP,1e5);
>> FIT = mamap22_fit_gamma_fs_trace(T,C);
>> mtrace_forward_moment(T,C,1) - mmap_forward_moment(FIT,1)
ans =
   1.1102e-16
  -2.8148e-06
```

Fit the trace with a higher-order MMAP matching the cross moments up to order 3, class probabilities and two-step class transition probabilities.

```
>> FIT = mmap_mixture_fit_trace(T,C)
FIT =
  [8x8 double]   [8x8 double]   [8x8 double]   [8x8 double]
```


MMAP = mmap_mixture_fit_mmap(MMAP)
MMAP = mmap_mixture_fit_trace(T, C)
MMAP = mmap_mixture_fit(PCC,CROSS)
PH = aph2_fit_map(MAP)
PH = aph2_fit_trace(T)
PH = aph2_fit(m1,m2,m3)
PHS = aph2_fitall(m1,m2,m3)
PH = aph2_assemble(l1, l2, p1)
[M2a,M3a] = aph2_adjust(M1,M2,M3)
MAPH = maph2m_fit_mmap(MMAP)
MAPH = mamap2m_fit_trace(T,C)
MAPH = maph2m_fit(m1,m2,m3,p,B)
MAPH = maph2m_fit_multiclass(APH,p,B)
MMAP = mamap2m_fit_gamma_fb_mmap(MMAP)
MMAP = mamap2m_fit_gamma_fb_trace(T,C)
MMAP = mamap2m_fit_gamma_fb(m1,m2,m3,gamma,p,F,B)
MMAP = mamap2m_fit_fb_multiclass(MAP,p,F,B)
MAP = amap2_fit_gamma_map(MAP)
MAP = amap2_fit_gamma_trace(T)
MAP = amap2_fit_gamma(m1,m2,m3,gamma)
MAPS = amap2_fitall_gamma(m1,m2,m3,gamma)
MAP = amap2_assemble(l1,l2,p1,p2,form)
[M2a,M3a,GAMMa] = amap2_adjust_gamma(M1,M2,M3,GAMMA)
MMAP = mamap22_fit_gamma_fs_mmap(MMAP)
MMAP = mamap22_fit_gamma_fs_trace(T,C)
MMAP = mamap22_fit_gamma_fs(m1,m2,m3,gamma,p,F,S)
MMAP = mamap22_fit_gamma_fs_multiclass(MMAP,p,F,S)
MMAP = mamap22_fit_gamma_fs_mmap(MMAP)
MMAP = mamap22_fit_gamma_bs_trace(T,C)
MMAP = mamap22_fit_gamma_bs(m1,m2,m3,gamma,p,B,S)
MMAP = mamap22_fit_gamma_bs_multiclass(MMAP,p,B,S)