Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks

Arnaud Doucet,
Cambridge University

Nando de Freitas, Kevin Murphy & Stuart Russell,
UC Berkeley

Outline

Dynamic Bayesian Networks (DBNs):
- ABC network

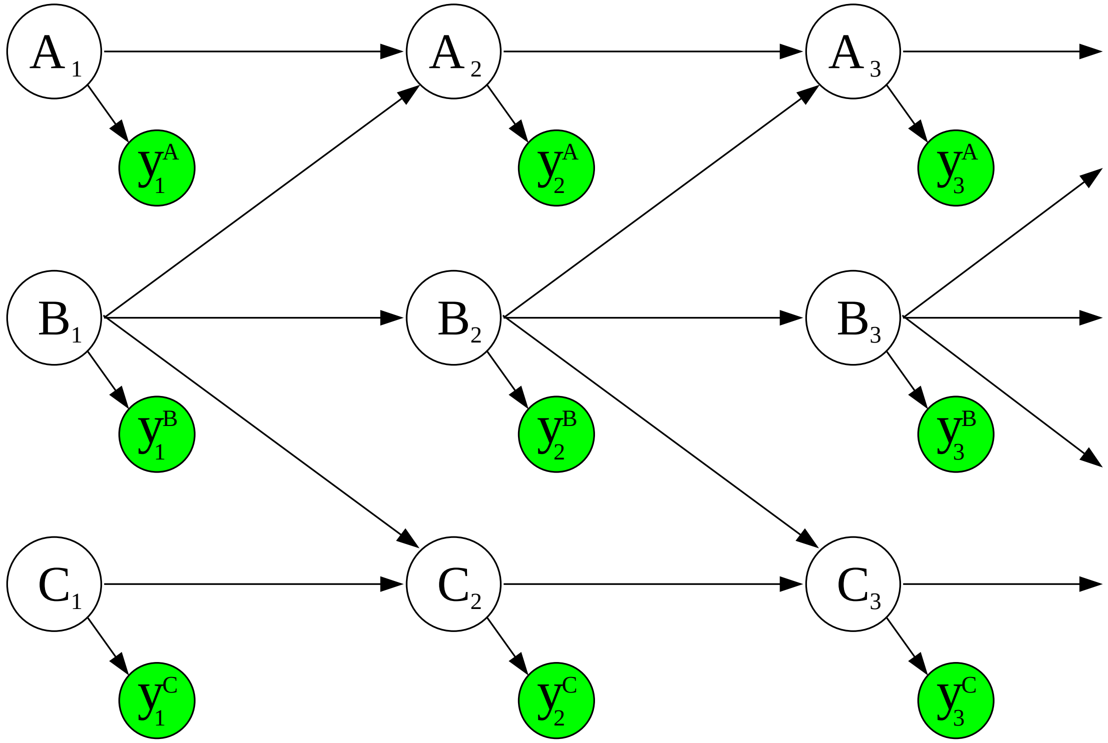
Inference in DBNs:
- Exact inference
- Particle filtering
- Rao-Blackwellised particle filtering

Demonstration on the ABC network

Results on large networks
- Map learning network
- Variable dimension radial basis network

Conclusions and further work

ABC network



Goal: compute the *joint filtering distribution* $p\left(A_t, B_t, C_t \middle| \mathbf{y_{1:t}}\right)$

Note the factorisation (using the notation $\mathbf{A_{1:t}} \triangleq \{\mathbf{A_1}, \mathbf{A_2}, \ldots, \mathbf{A_t}\}$):

$$\begin{aligned} p\left(\mathbf{A_{1:t}}, \mathbf{B_{1:t}}, \mathbf{C_{1:t}} \middle| \mathbf{y_{1:t}}\right) &= p\left(\mathbf{A_{1:t}}, \mathbf{C_{1:t}} \middle| \mathbf{y_{1:t}}, \mathbf{B_{1:t}}\right) p\left(\mathbf{B_{1:t}} \middle| \mathbf{y_{1:t}}\right) \\ &= p\left(\mathbf{A_{1:t}} \middle| \mathbf{y^{A}_{1:t}}, \mathbf{B_{1:t-1}}\right) p\left(\mathbf{C_{1:t}} \middle| \mathbf{y^{C}_{1:t}}, \mathbf{B_{1:t-1}}\right) p\left(\mathbf{B_{1:t}} \middle| \mathbf{y_{1:t}}\right) \end{aligned}$$

Inference methods for DBNs

Exact:

– $O(q^{2\times 3})$ operations per time step for discrete nodes with q values.

– Impossible for almost all continuous distributions.

Particle filtering (PF):

– Monte Carlo approximation algorithm that applies to any distribution.

– $O(N)$ operations per time step. N is the number of samples.

– Estimates can have high variance.

Rao Blackwellised particle filtering (RBPF):

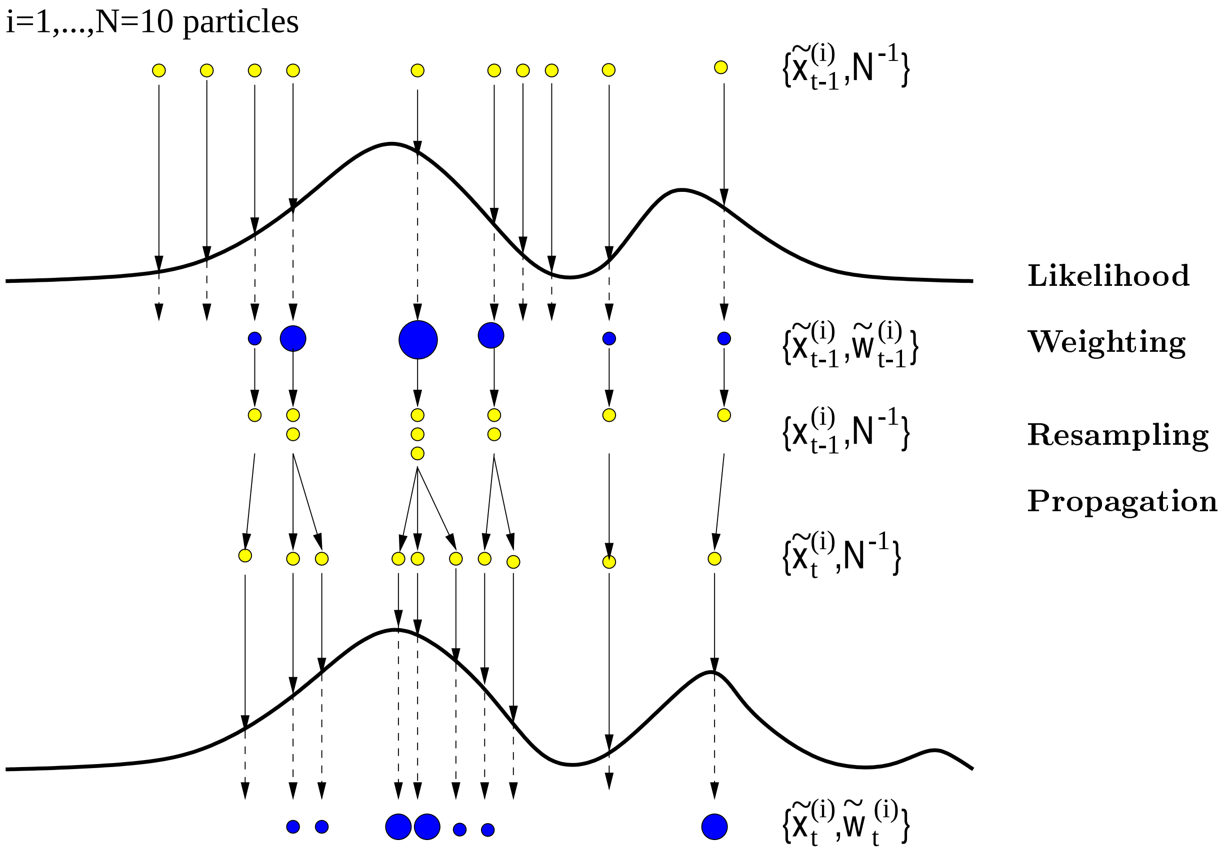
– Sample a subset of the variables and compute the distributions of the remaining variables exactly.

– Reduces the variance problem (Casella and Robert, 1996).

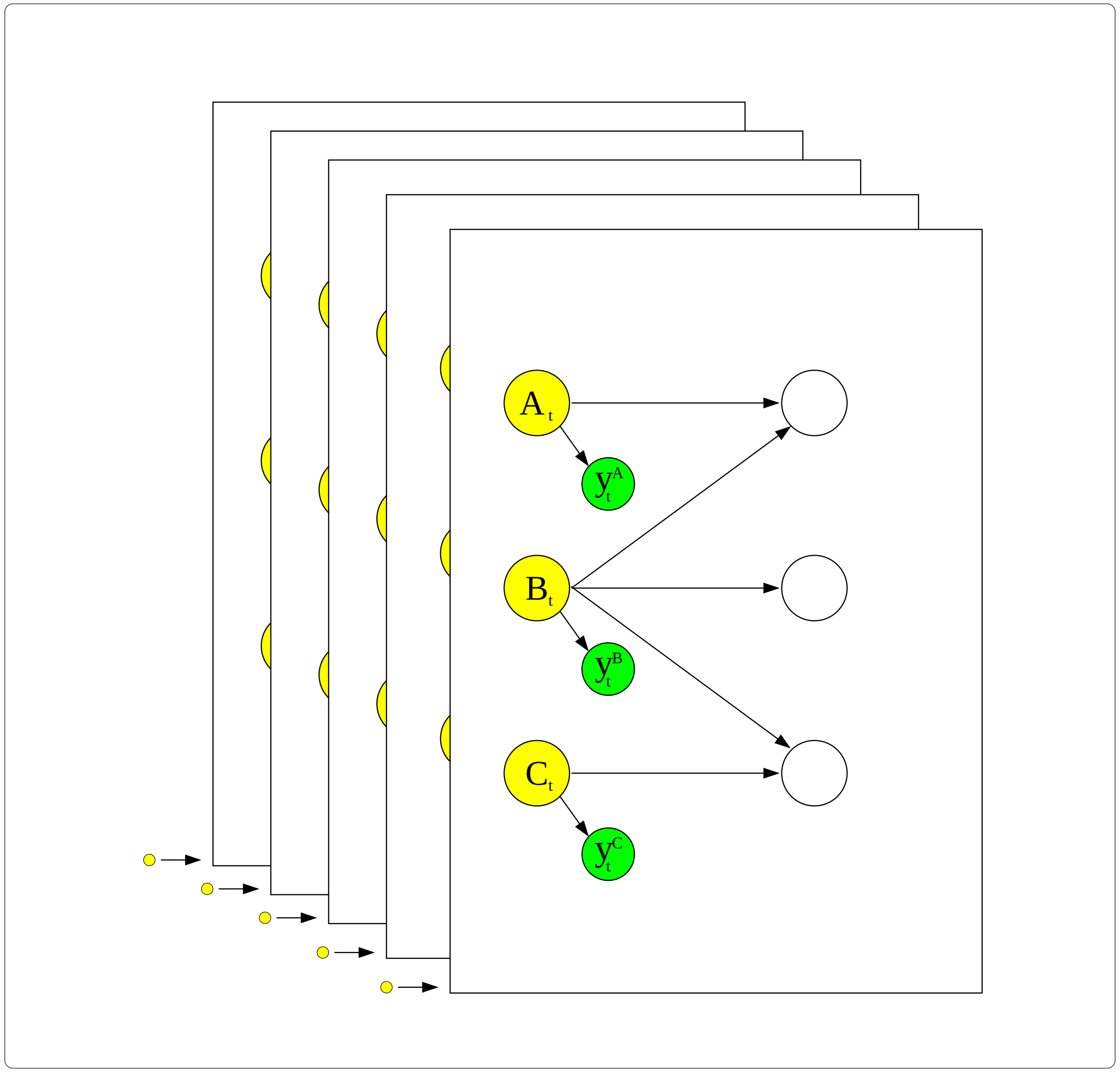
– $O(q^2 N')$ operations per time step for the ABC network.

Particle filtering

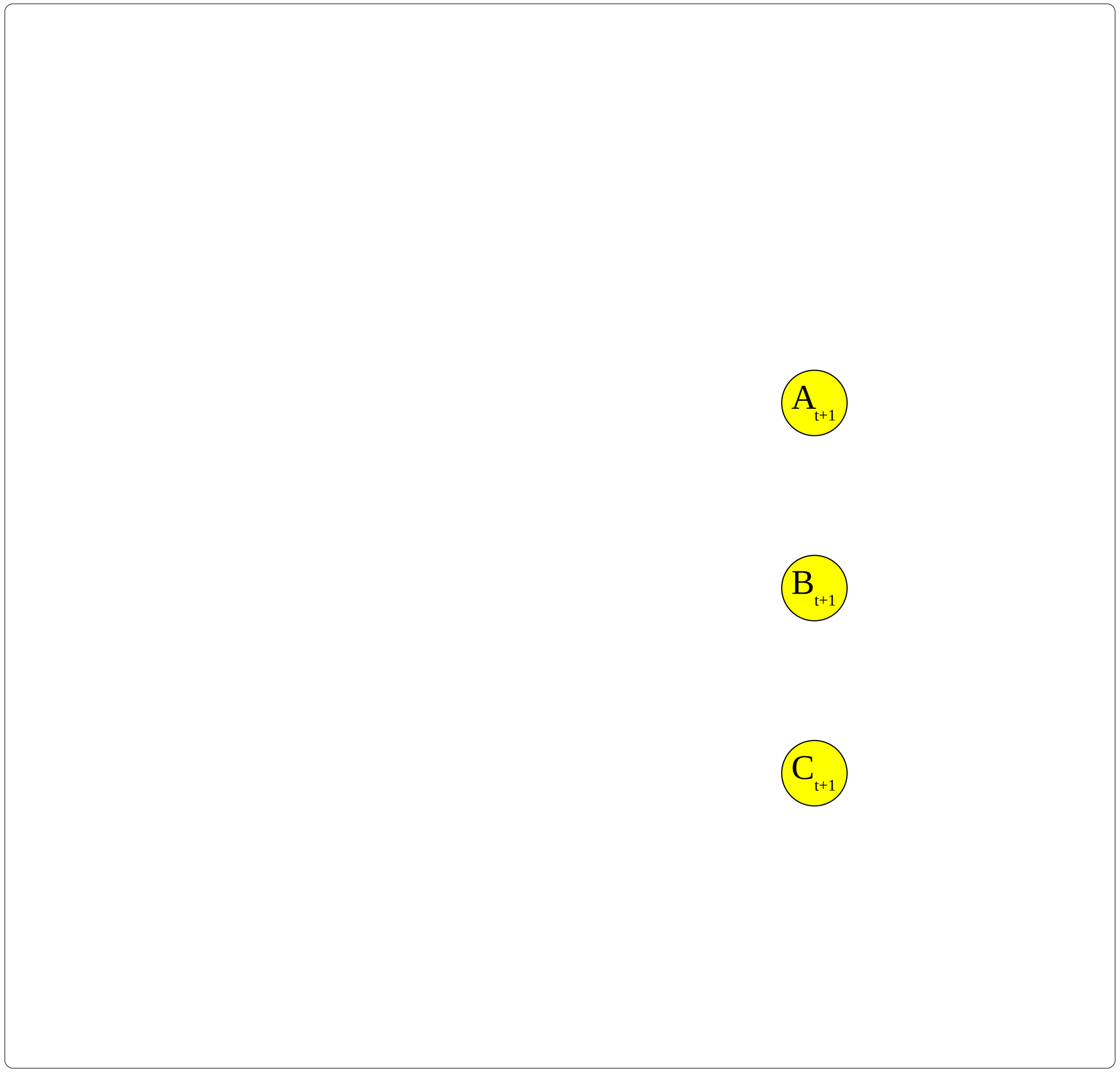
Idea: approximate the filtering distribution with a finite set of samples (particles). Then compute estimates by Monte Carlo averaging.



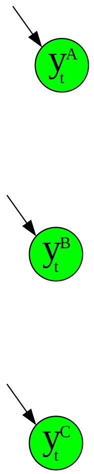
PF for the ABC network



PF for the ABC network



PF for the ABC network



Rao-Blackwellised particle filtering

Problem: PF can be inefficient if the state space is large.

Solution: *Rao-Blackwellisation*, that is, sample a subset of the variables allowing the remainder to be integrated out exactly. The resulting estimates can be shown to have lower variance.

Rao Blackwell Theorem: *Let a DBN have two groups of variables,* $\mathbf{R}$ *and* $\mathbf{L}$. *Then, for the joint distribution* $\pi(\mathbf{R}, \mathbf{L})$, *the following result applies*

$$\mathrm{Var}_{\pi}\left[\mathbb{E}_{\pi}\{f(\mathbf{R})|\mathbf{L}\}\right] \leq \mathrm{Var}_{\pi}\left[f(\mathbf{R})\right]$$

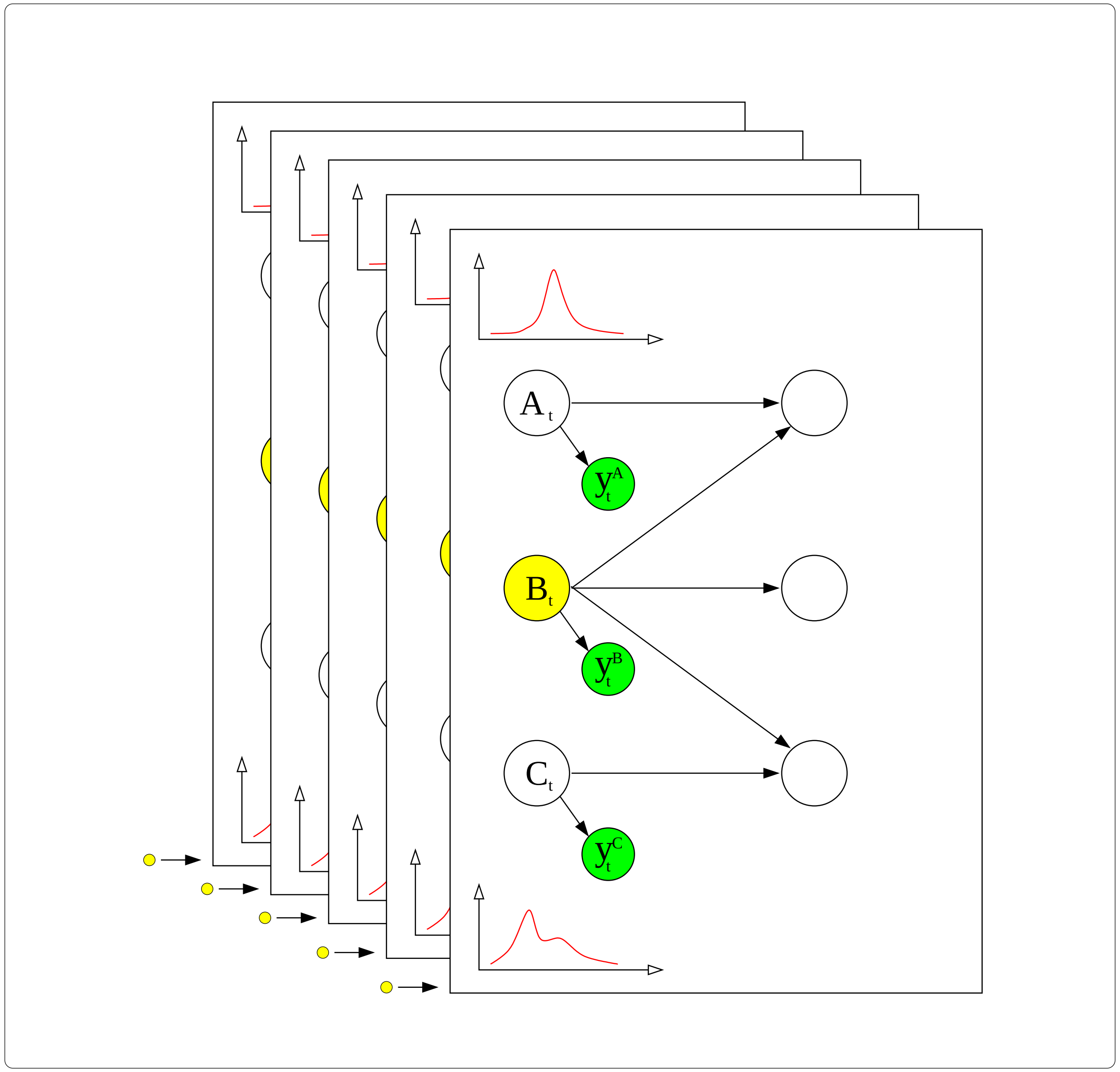
for a function of interest f, *e.g. the mean or covariance (Casella & Robert, 1996, Liu et. al. 1995).*

Rao-Blackwellised particle filtering

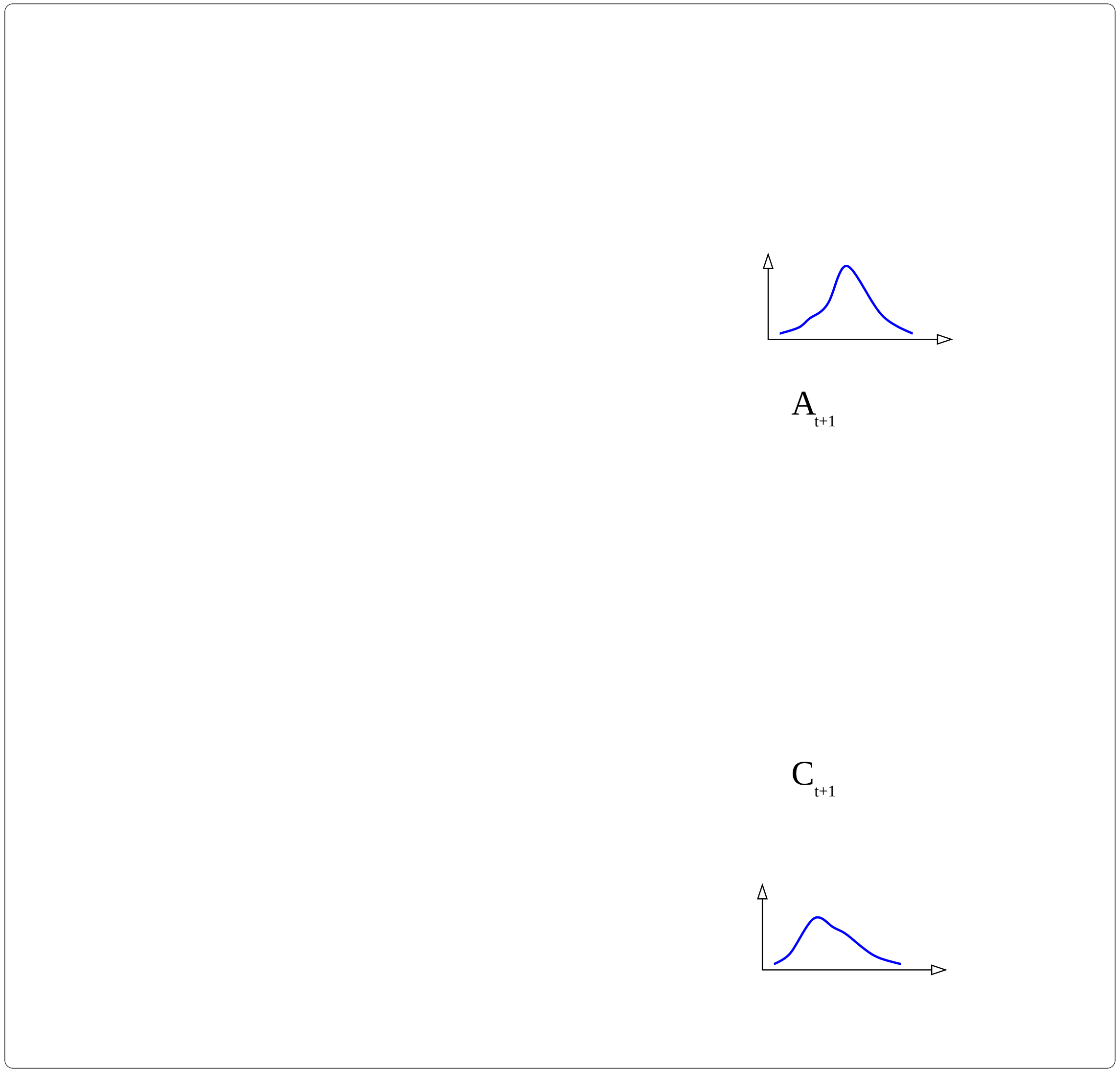
ABC network: Sample root node $\mathbf{B}_{1:t}$ and then compute the distributions of the leaves $p(A_t|\mathbf{B}_{1:t}, \mathbf{y}_{1:t})$ and $p(C_t|\mathbf{B}_{1:t}, \mathbf{y}_{1:t})$ using the exact HMM filter (junction tree algorithm).

General networks: Any network admitting a factored representation where one of the marginals can be computed with a finite dimensional filter.

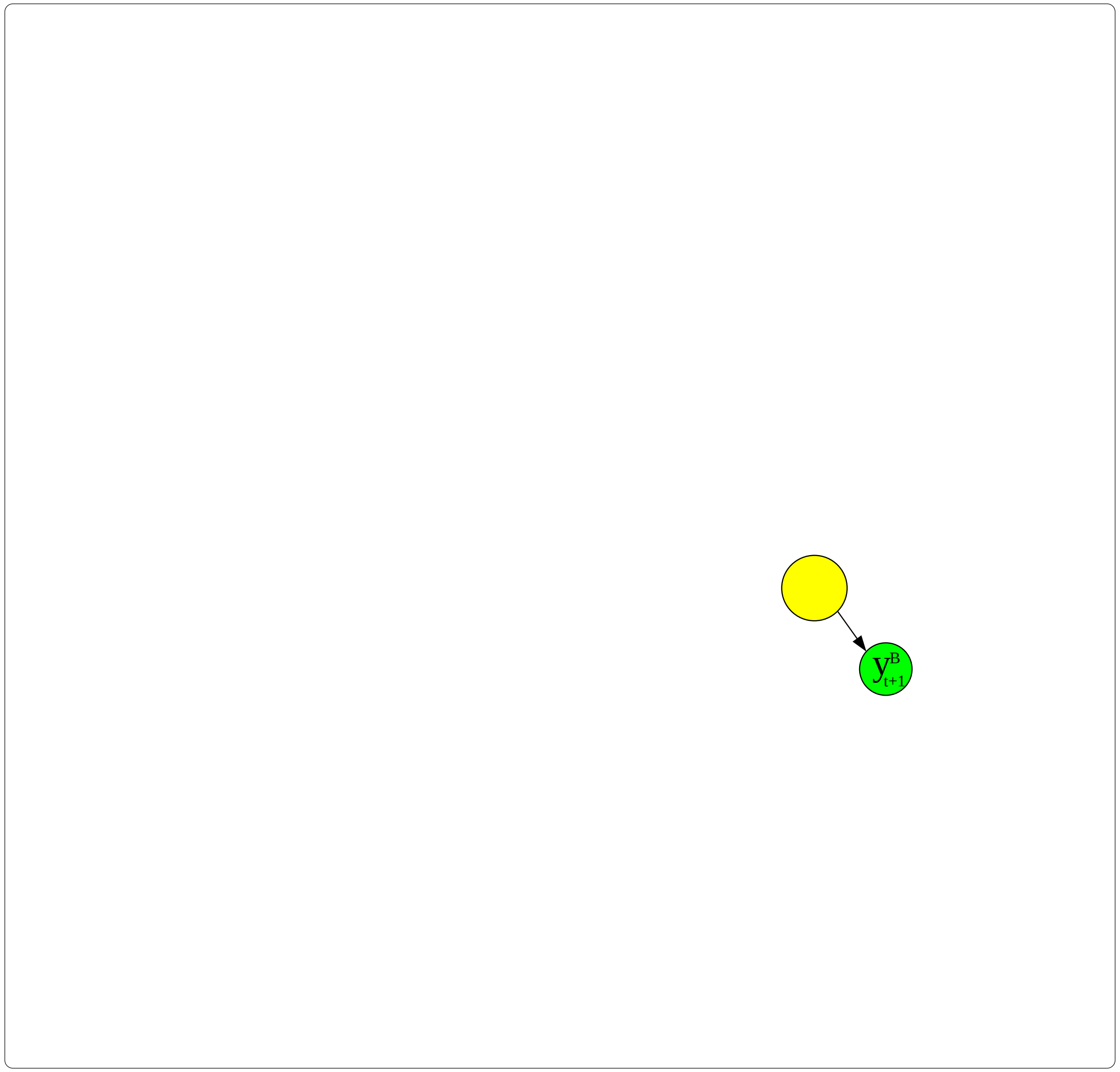
RBPF for the ABC network



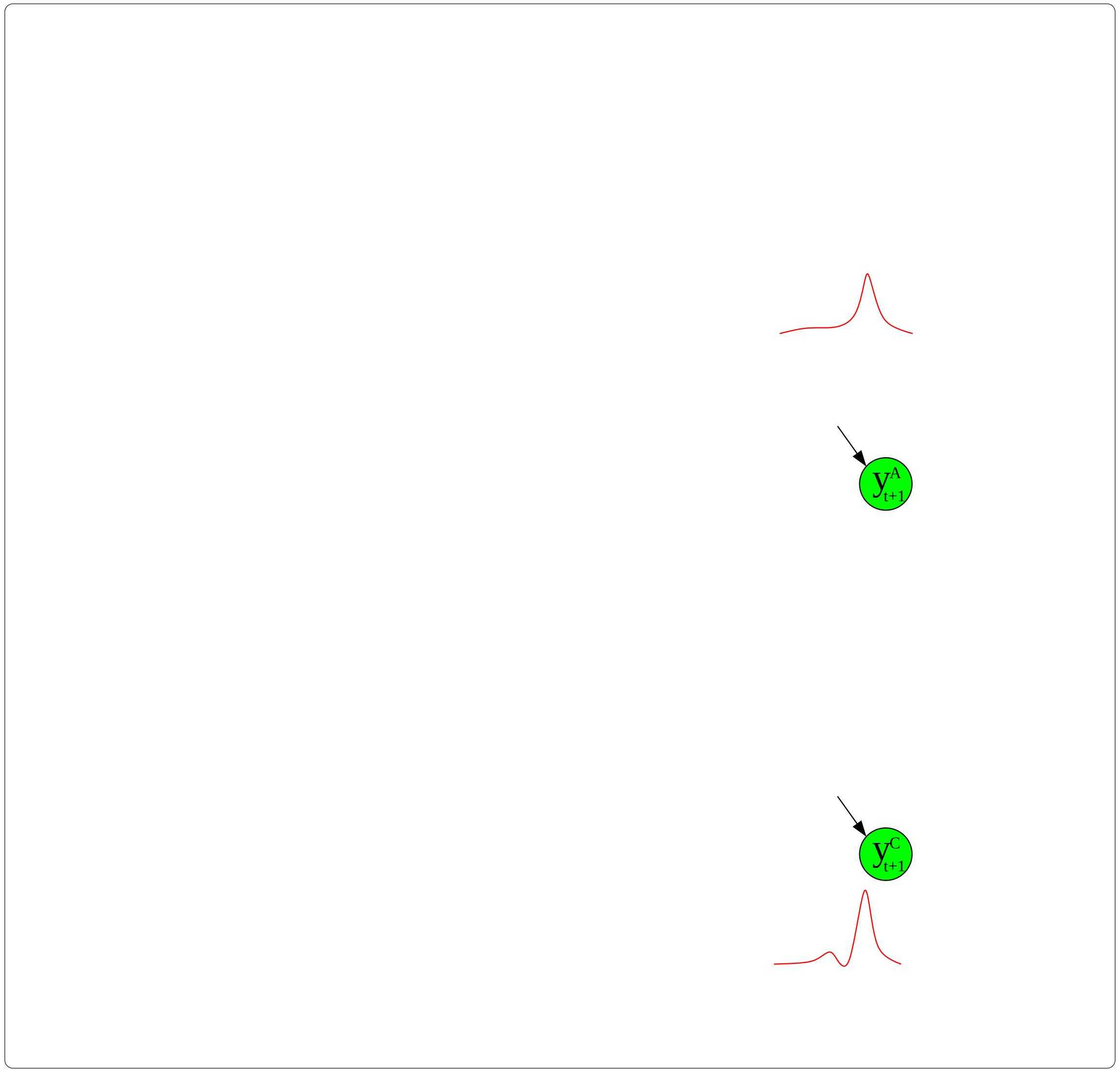
RBPF for the ABC network



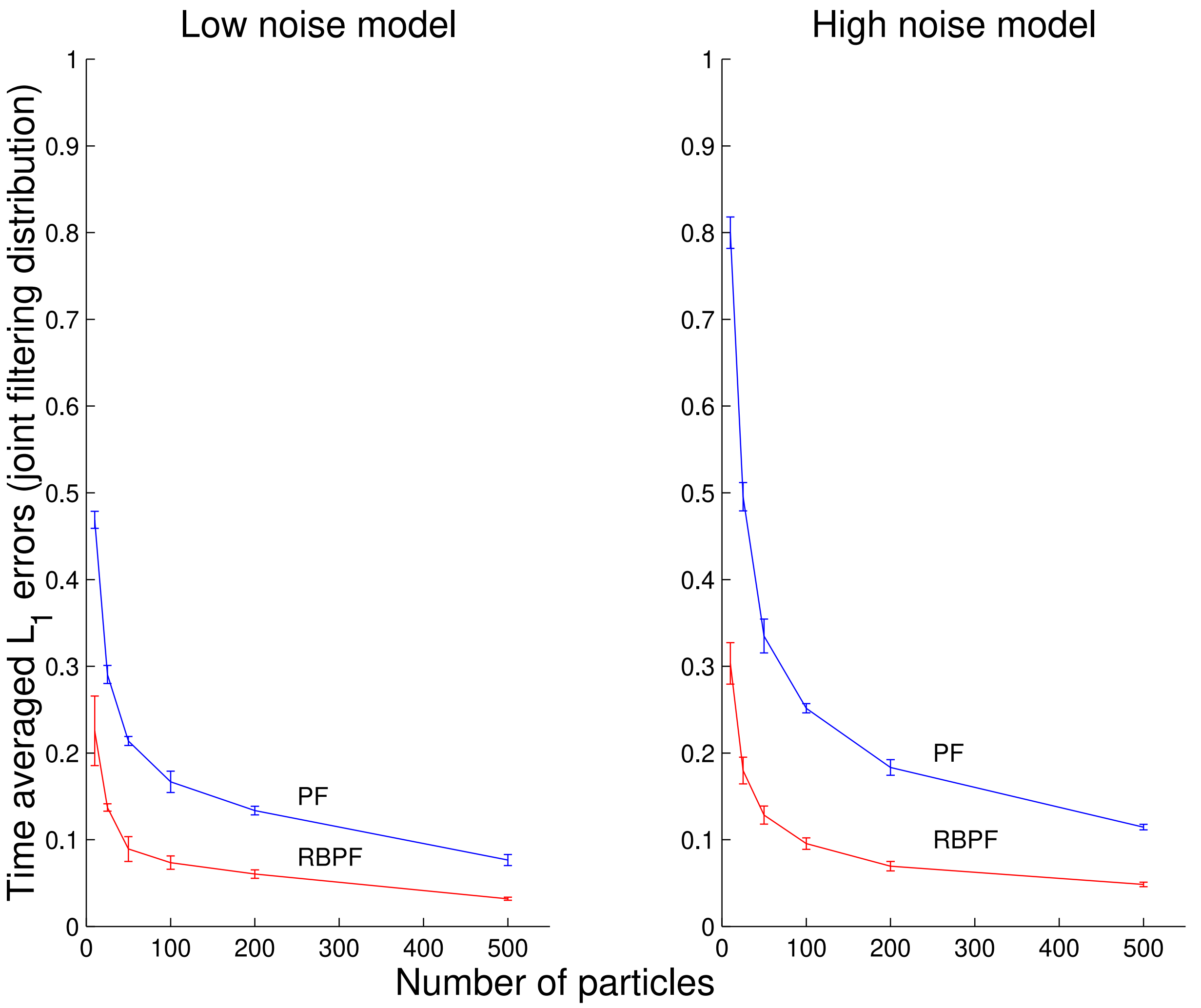
RBPF for the ABC network



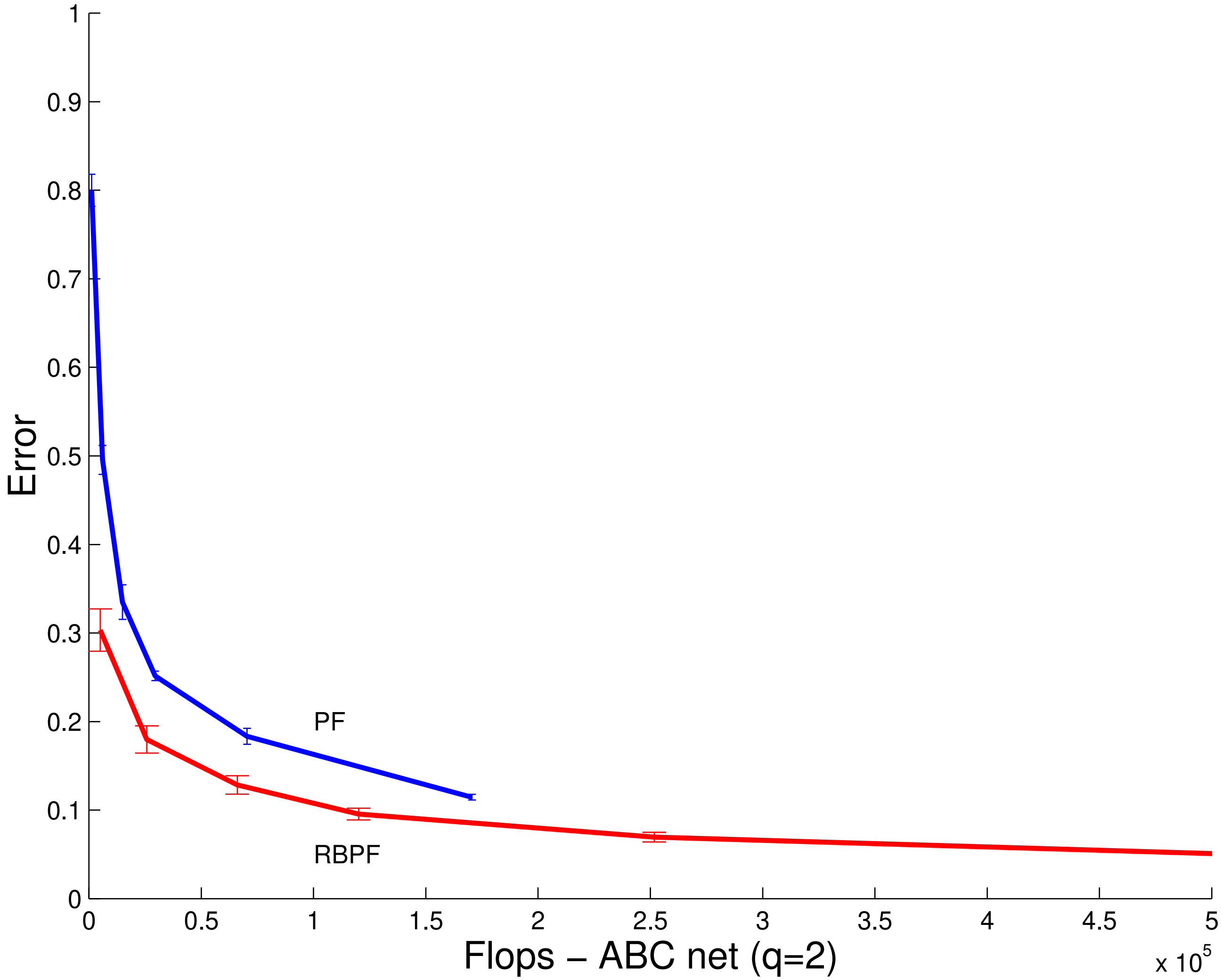
RBPF for the ABC network



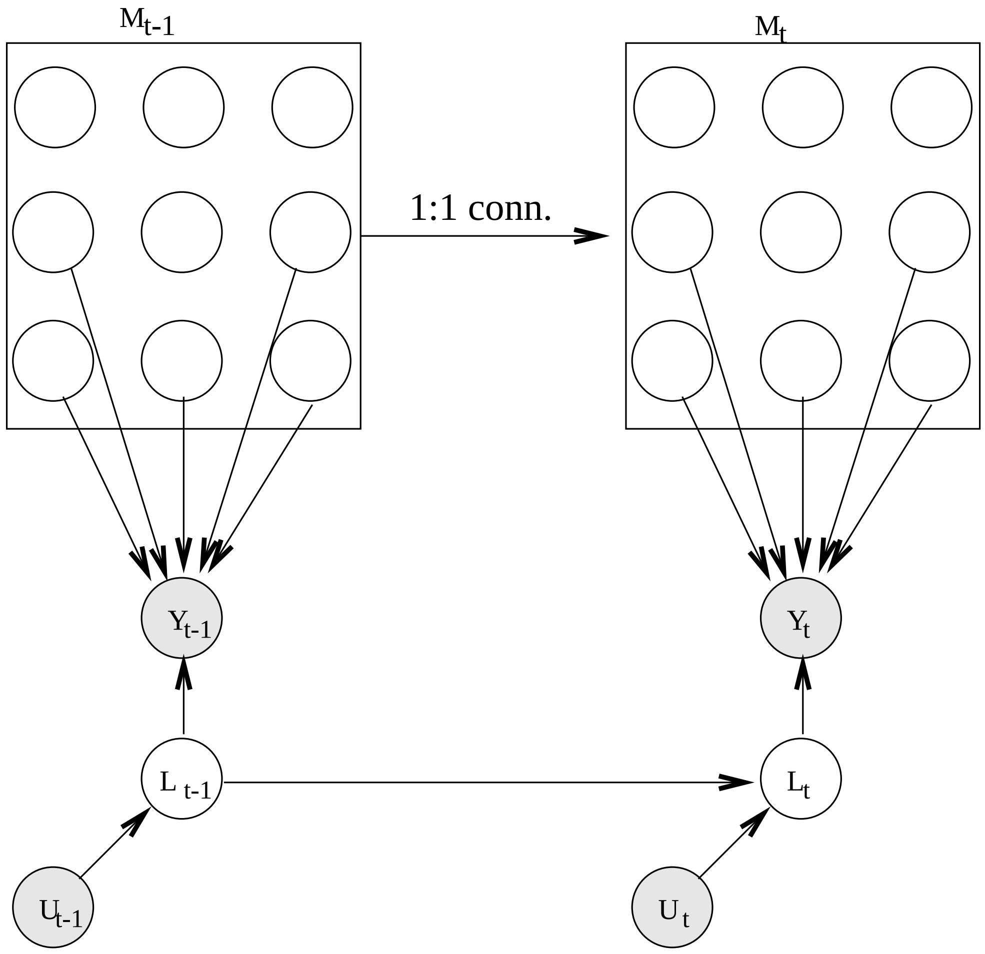
Experiment with the ABC network



Experiment with the ABC network



Map learning network



L_t = robot location at time t

$Y_t = 3 \times 3$ observation (noisy)

$M_t(i)$ = contents of cell i at t

U_t = Control at t (noisy actuators)

Map learning network

The goal of concurrent *robot localisation* and *map learning* is to compute the distribution $P(L_t, M_t|Y_{1:t}, U_{1:t})$.

For an occupancy grid (map with binary cells, $q = 2$) of size $M \times M$, the state space has dimension dimension 2^{M^2}. For a 10×10 grid we need 1.6×10^{60} operations !!!!!!!

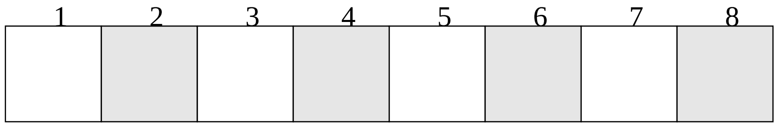
Note the factorisation:

$$p(M_t, L_{1:t}|U_{1:t}, Y_{1:t}) = p(L_{1:t}|U_{1:t}, Y_{1:t})p(M_t|U_{1:t}, Y_{1:t}, L_{1:t})$$

Key idea: Once we know the current location L_t, it is easy to compute the map distribution $P(M_t|U_{1:t}, Y_{1:t}, L_{1:t})$.

Experiments with the map learning network

1×8 grid world, each cell is either black or white.

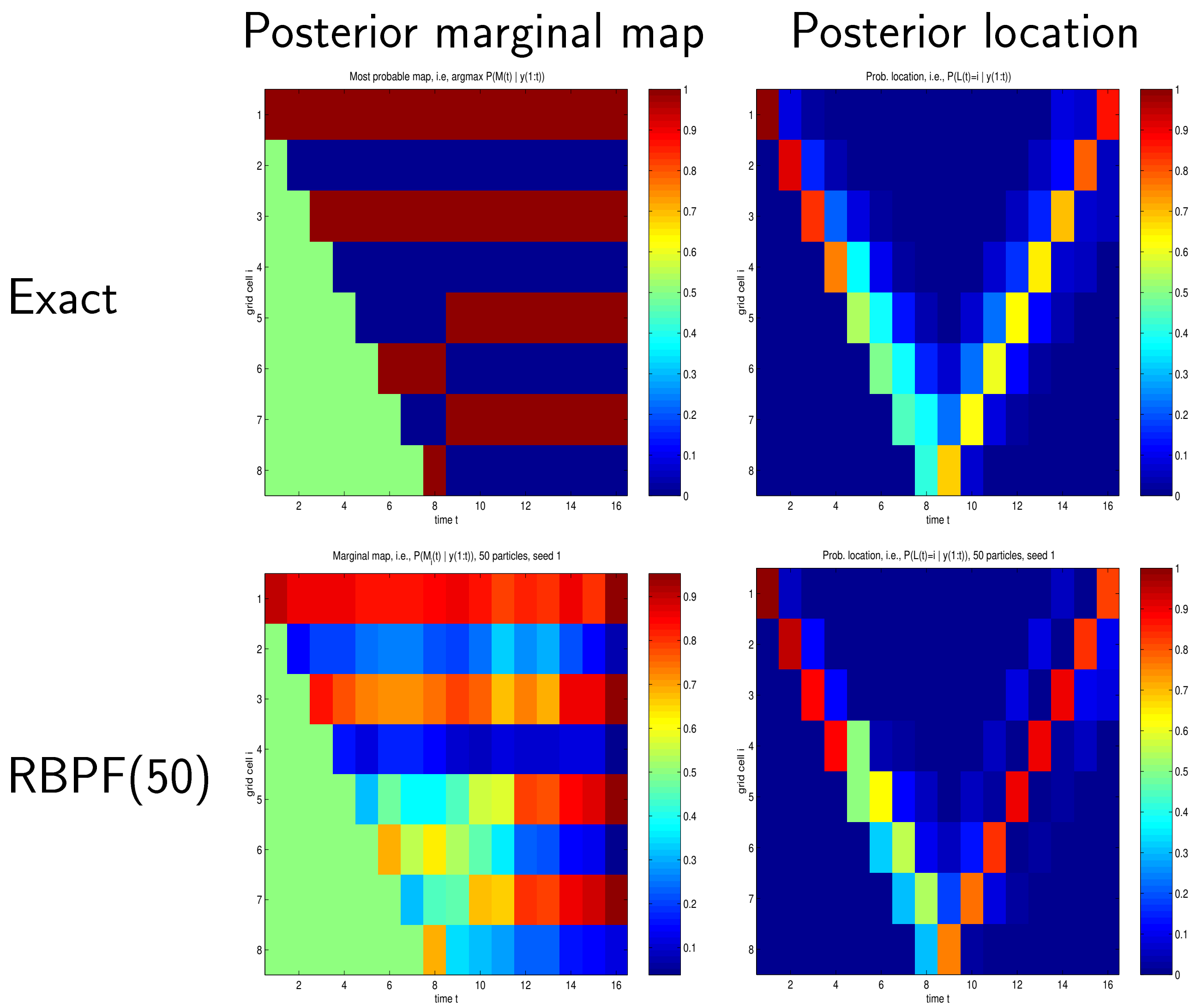


Exact inference takes about 50,000 operations per timestep (!).

Trajectory: left–right–left, but slips once at $t = 4$.

t	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
L_t	1	2	3	**4**	**4**	5	6	7	8	7	6	5	4	3	2	1
Y_t	0	1	0	1	1	0	1	0	1	0	1	0	1	0	1	0
U_t	-	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\rightarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$	$\leftarrow$

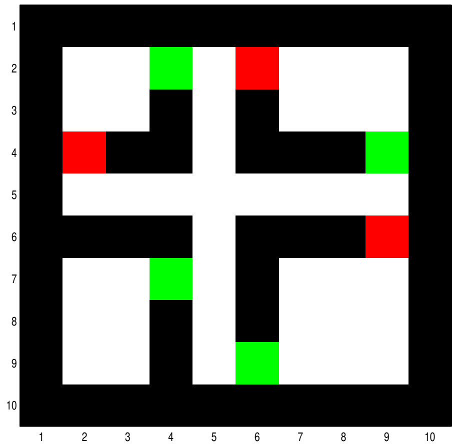
Exact vs. RBPF

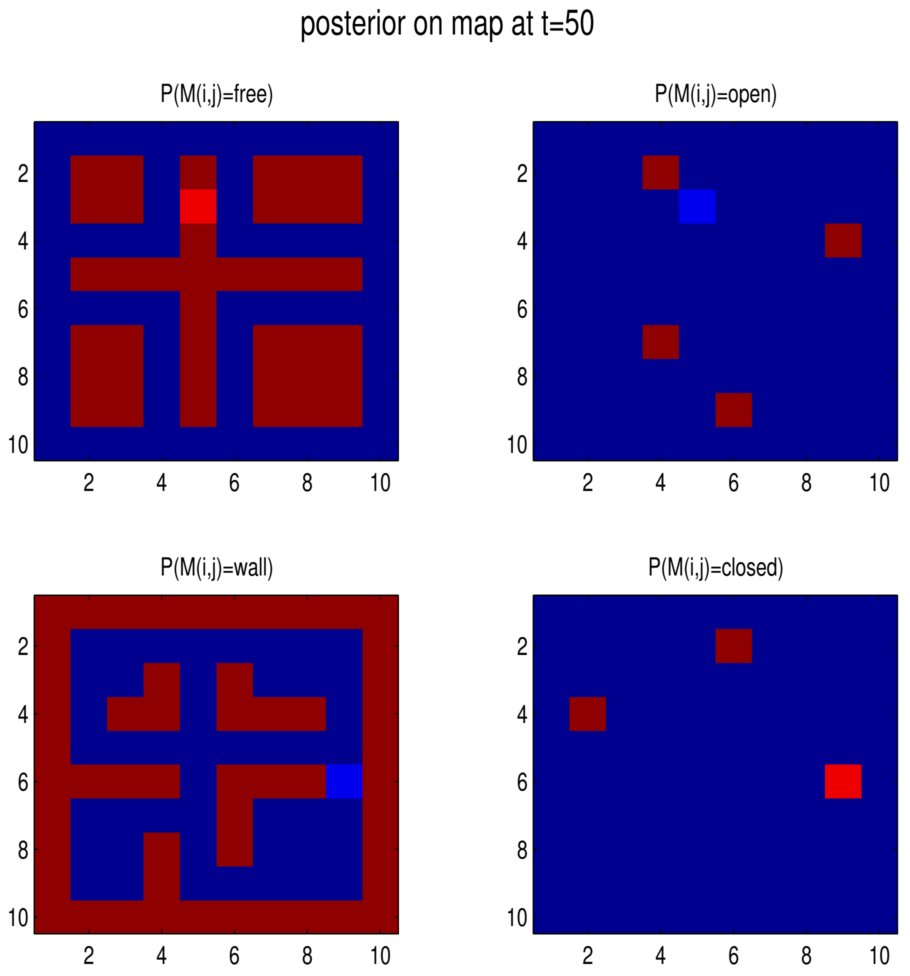


A 2D map example

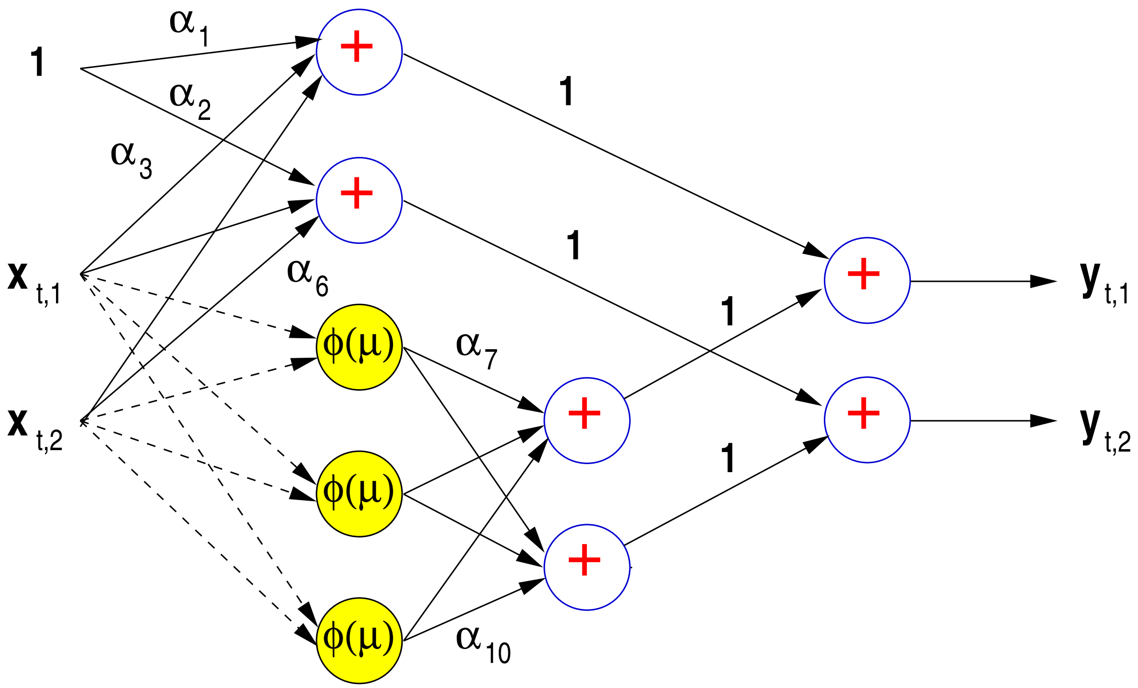
Self-guided exploration; doors can open or close w.p. 0.1

Rao-Blackwellized particle filtering, 100 particles, 50 steps





Time-varying radial basis network



This network represents a variable dimension generalised linear model:

$$\mathbf{y_t} = \mathbf{f_t}(\mathbf{k_t}, \mu_\mathbf{t}^{(\mathbf{k})}, \mathbf{x_t})\alpha_\mathbf{t}^{(\mathbf{k})} + \mathcal{N}(\mathbf{0}, \sigma^2\mathbf{I})$$

$\mathbf{x_t}$ = Input data　　$\mathbf{y_t}$ = Output data

$\mu_\mathbf{t}$ = Basis locations　　$\alpha_\mathbf{t}$ = Basis coefficients

k_t = Model order　　$\sigma_\mathbf{t}^2$ = Noise variance

$\mathbf{f}$ = Nonlinear, non-stationary multivariate function.

Time-varying radial basis network

The goal is to compute $p(k_t, \mu_{\mathbf{t}}^{(\mathbf{k})}, \alpha_{\mathbf{t}}^{(\mathbf{k})}, \sigma_{\mathbf{t}}^{\mathbf{2}} | \mathbf{x}_{\mathbf{1:t}}, \mathbf{y}_{\mathbf{1:t}})$ to solve problems such as time series analysis (speech processing, financial forecasting), regression (control) and classification (fault detection).

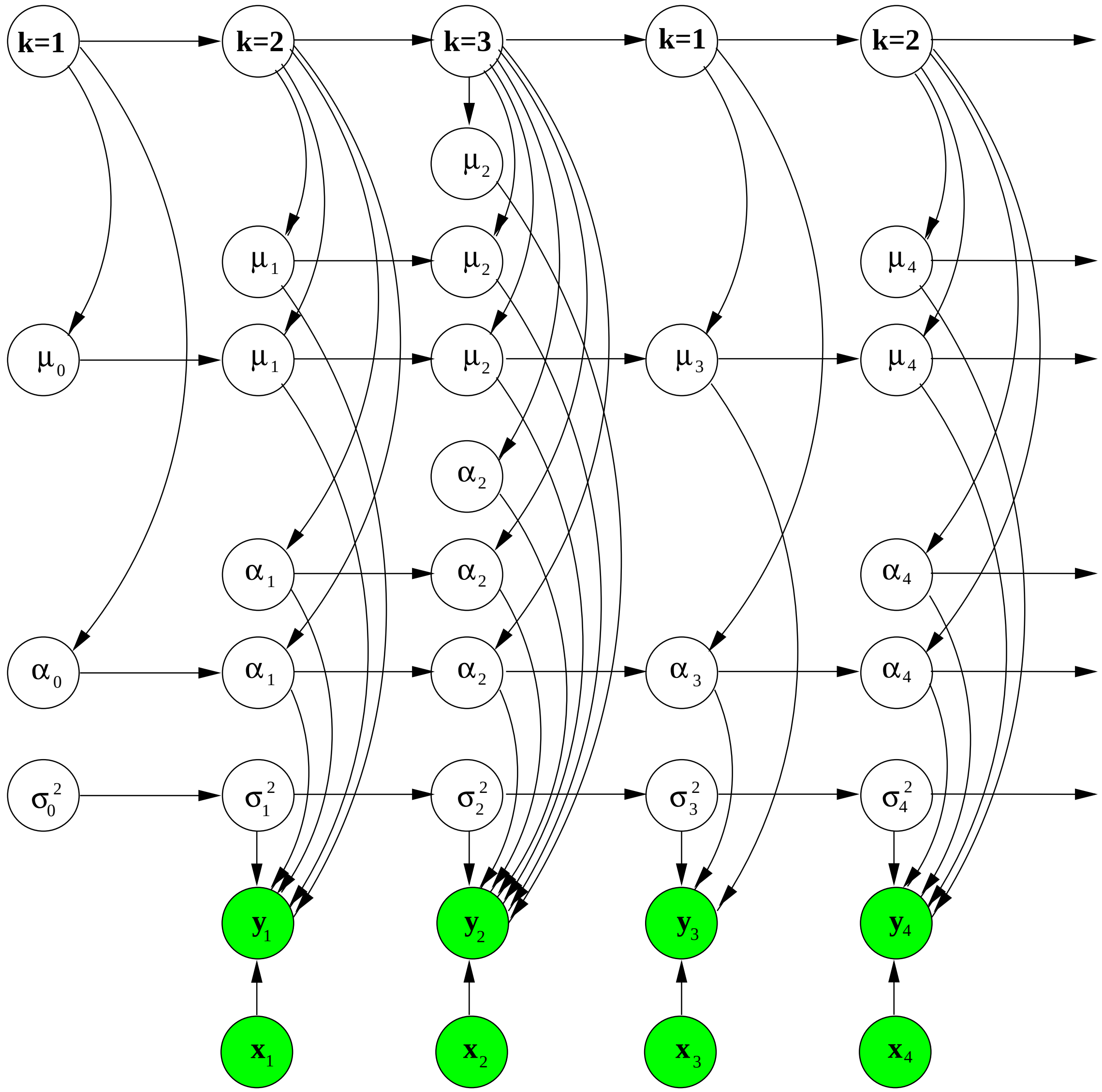
Difficult problem involving *model selection/averaging*, *parameter estimation* and *mixed continuous and discrete states*.

Note the factorisation:

$$p(k_t, \mu_{\mathbf{t}}^{(\mathbf{k})}, \alpha_{\mathbf{t}}^{(\mathbf{k})}, \sigma_{\mathbf{t}}^{\mathbf{2}} | \mathbf{x}_{\mathbf{1:t}}, \mathbf{y}_{\mathbf{1:t}}) = \\ p(\alpha_{\mathbf{t}}^{(\mathbf{k})} | \mathbf{k}_{\mathbf{t}}, \mu_{\mathbf{t}}^{(\mathbf{k})}, \sigma_{\mathbf{t}}^{\mathbf{2}}, \mathbf{x}_{\mathbf{1:t}}, \mathbf{y}_{\mathbf{1:t}}) \mathbf{p}(\mathbf{k}_{\mathbf{t}}, \mu_{\mathbf{t}}^{(\mathbf{k})}, \sigma_{\mathbf{t}}^{\mathbf{2}} | \mathbf{x}_{\mathbf{1:t}}, \mathbf{y}_{\mathbf{1:t}})$$

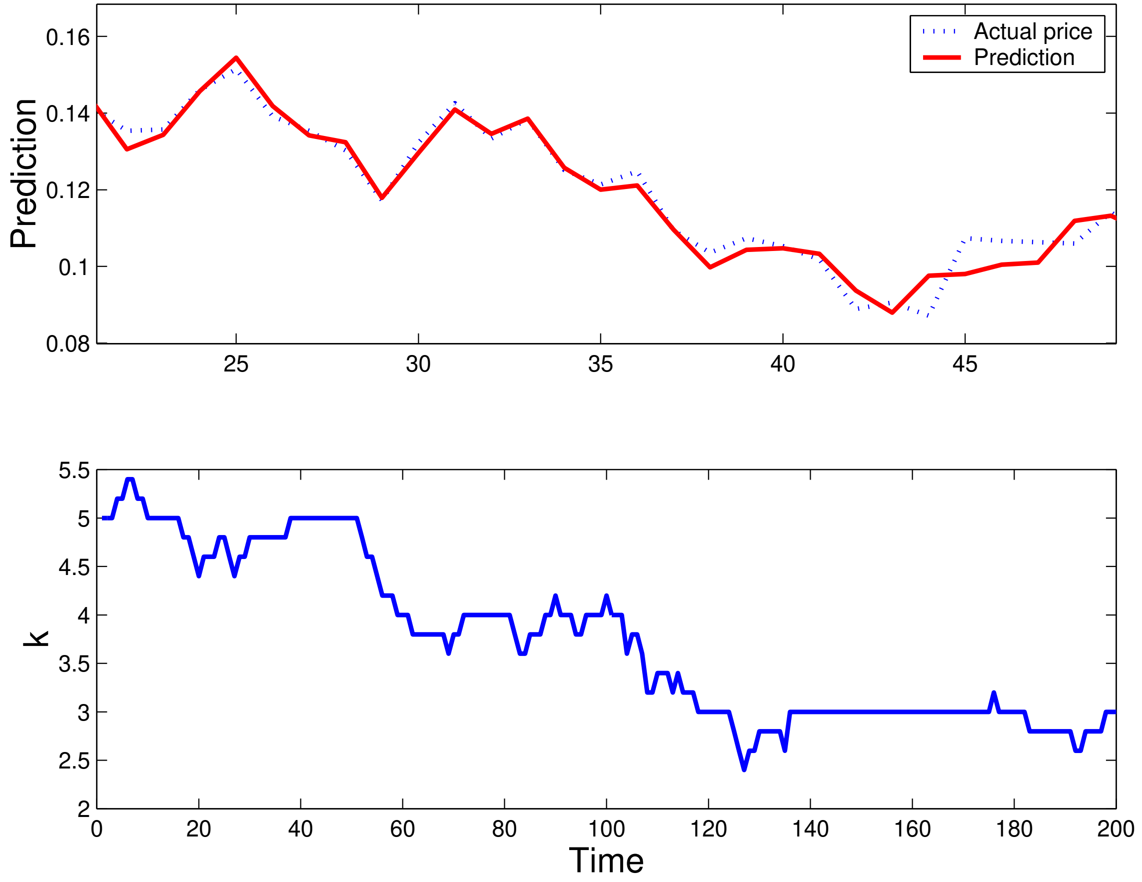
Sample $k_{1:t}, \mu_{\mathbf{1:t}}^{(\mathbf{k})}, \sigma_{\mathbf{1:t}}^{\mathbf{2}}$ and then compute the distribution of $\alpha_{\mathbf{t}}^{(\mathbf{k})}$ exactly using the Kalman filter algorithm.

Time-varying radial basis network



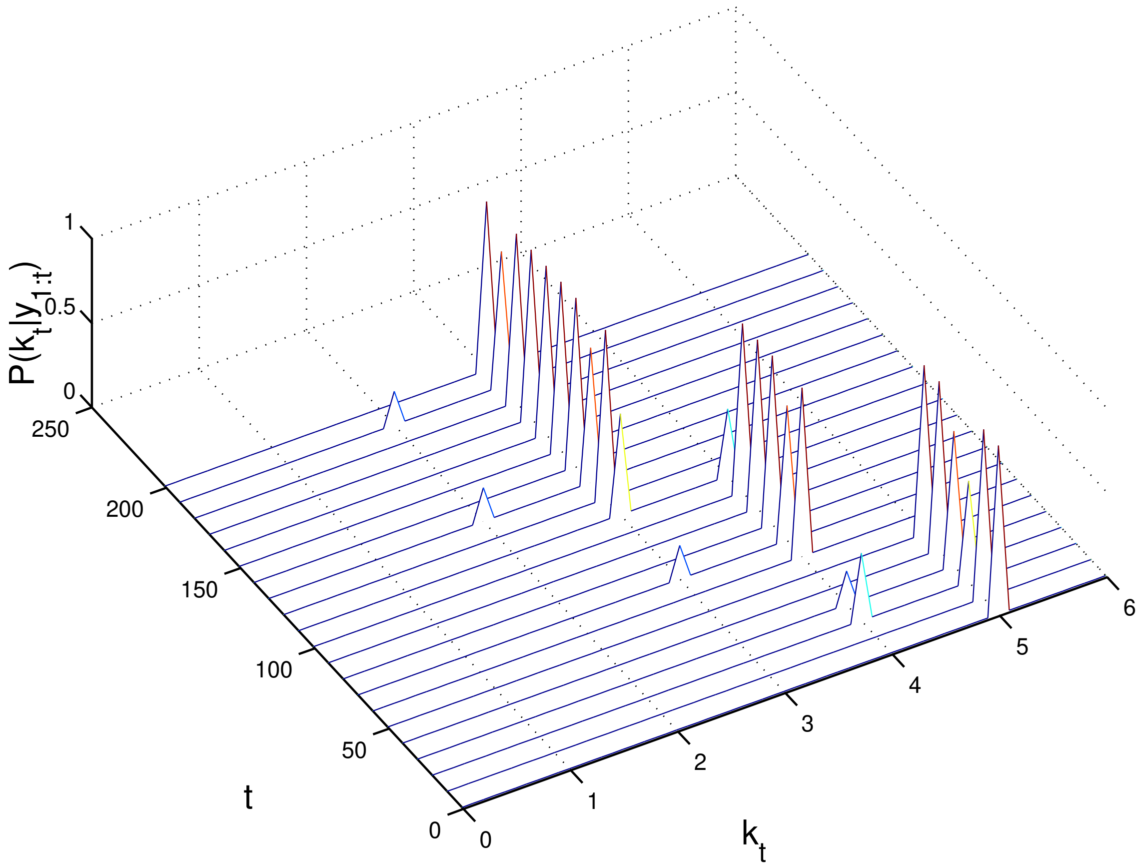
Forecasting financial instruments

Accurate modelling of real options data:



Forecasting financial instruments

Entire distributions for all quantities:



Conclusions and further work

RBPF works well for some large DBN models.

Further research on sparsely structured complex systems.

Automatic algorithms for general DBN topologies. That is, choose roots and leaves automatically.

Available material at `http://www.cs.berkeley.edu/~jfgf`

- Technical reports
- Tutorial report for the ABC network
- Matlab software for the ABC and RBF network
- Book information
- Particle filtering site

Exact vs. Boyen-Koller

Posterior marginal map — Posterior location

Exact

BK