About the time and space complexities of SMMB

As for the other algorithms dealing with Markov blanket learning, it is impossible to provide a theoretical time complexity for SMMB. The reason lies in the fact that the number of iterations in Algorithm 2 (repeat … until the MB is not modified) cannot be estimated. This number specifically depends on the data. It is not even possible to calculate a worst case time complexity because it is impossible to provide an upper bound for the number of iterations in Algorithm 2. However, it is known that a current bottleneck of SMMB is the time complexity of function *buildConsensus*. The backward phase in function learnMB is at worst $O(q\ 2^q)$, with q, the maximal size of a Markov blanket output by learnMB. Empirical feedback from our experiments indicates that q is lower than 4-5 and the complexity of the backward phase is not an issue in function learnMB. In contrast, the backward phase in function *buildConsensus* handles the union of all Markov blankets that have been generated. In this case, q may be up to a few hundreds. Nonetheless, in practice, it is observed that the size of the conditioning set (*i.e.* the MB consensus) rapidly decreases: generally, the first conditional test performed for a variable indicates that this variable must be discarded from the consensus MB. Thus the empirical complexity is instead close to $O(q\ e)$, where e denotes the complexity of the tests run on all permutations. A substantial part of the computational burden of *buildConsensus* is due to the correction for multiple tests, which is performed through permutations. Even if less time consuming *adaptive* permutations are run, the overall running time remains high. Our observations show that function *buildConsensus* runs for roughly 50%-60% of the total running time.

Regarding the theoretical space complexity of SMMB, at most r sub-optimal Markov blankets are built. An upper bound for this complexity is then $O(r\ q)$, with q, the maximal size of a Markov blanket output by learnMB. In function buidConsensus, correction for multiple testing relies on permutations. Vectors of permuted phenotypes are prepared and stored at the beginning of the execution of buildConsensus. If the number of permutations is n_p (1000 in our case), the memory used for this purpose scales in $O(n_p\ n)$, where n is the number of observations (*i.e.* individuals).

Empirical time complexities are compared below for SMMB, BEAM, DASSO-MB and AntEpiSeeker on the simulated dataset relative to Model 1 (Multiplicative model) (see Section 4.1. (Simulated datasets)).

Table 1. Comparison of running times for SMMB, BEAM, DASSO-MB and AntEpiSeeker on simulated data (2,000 cases and 2,000 controls; 100 SNPs), for the multiplicative model (Model 1, see 4.1. (Simulated datasets)). XEON biprocessors 5462 2.66 GHz, 6 cores.

Software	**SMMB**	**BEAM**	**DASSO-MB**	**AntEpiSeeker**

Parameters	t = 1000
	r = 100
	m = 30
	K = 10
	k = 3
	α = 0.05
		nb iterations for burn-in phase: 1000
		nb iterations for stationary phase:10000
			α = 0.05	450 iterations
				1000 ants
				α = 0.01

Average running time over 100 executions (s)	30 s	93 s	5s	469s