We thank the reviewers for their comments. As requested by R1/R3 we first report an empirical comparison with previous work. We then address reviewer's comments individually (due to space limits please zoom in the tiny figures).

ε	B_ε		Time (s)		Tuning Time (s)	
	Ours	AM [12]	**Ours**	AM [12]	**Ours**	AM [12]
.01	$.476 \pm \sigma$	$.477 \pm \sigma$	13 ± 1	12 ± 1	–	482 ± 20
.005	$.434 \pm \sigma$	$.436 \pm \sigma$	16 ± 1	15 ± 1	–	557 ± 14
.001	$.388 \pm \sigma$	$.401 \pm \sigma$	20 ± 2	7 ± 2	–	242 ± 8

Table 1: Comparison of our algorithm and [12] on Ellipses (same setting in the paper). $\sigma < 10^{-4}$.

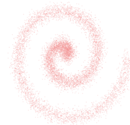

Figure 1

m	n: 100	n: 1000
10	29$\pm$ 4 s	33 ± 6 s
50	8$\pm$ 1 min	9 ± 1 min
100	15 $\pm$1 min	24 ± 2 min

Table 2: Time to reach relative improvement 10^{-4} of B_ε ($\varepsilon = 0.01$).

Comparison: we focus on [12,18] since they consider entropic regularization similarly to our setting. Since no code was available, experiments are based on our implementations of [12,18] (run on a Nvidia Tesla M40).

Alternating Minimization (AM) [12]: AM iteratively optimizes the support points and weights of the barycenter. Table 1 reports the value of the objective functional B_ε and the running times (run until relative improvement of B_ε was $< 10^{-3}$) on the "30 ellipses dataset" (see our paper) for $\varepsilon = 0.01, 0.005, 0.001$. We note that while a single run of AM is slightly faster, it exhibits worse performance (in particular as ε decreases). Moreover, differently from our method, optimization in [12] requires tuning multiple parameters (e.g. step-size), leading to significantly longer times. We run AM with a budget of 500 support points. Our method stops at ~ 300 support points.

Decentralize Barycenters [18]: similarly to our method, [18] can compute barycenters of continuous measures (via sampling), *but* the barycenter's support points are *fixed a priori* and only the weights are computed. Moreover since [18] minimizes a different objective functional (it does not consider the *unbiased* formulation of the Sinkhorn divergence), here we focus on a qualitative comparison. We compute the barycenter of 5 bidimensional Gaussian measures $\mathcal{N}(\mathsf{m}, \sigma^2 \mathrm{Id}_2)$, with m randomly sampled in $[0,1]^2$ and σ^2 in $[0,1]$. We set $\varepsilon = 0.01$ for both methods. For [18] we used Alg. 2 with complete agents' graph and a 50×50 support grid in $[0,1]^2$. Fig. 2 (left) shows how barycenters evolve over time. Our method appears to better capture the properties of the target barycenter, converging faster towards its solution. This is also reflected by the decreasing rate of the two corresponding measure of convergence for the two methods Fig. 2 (Right).

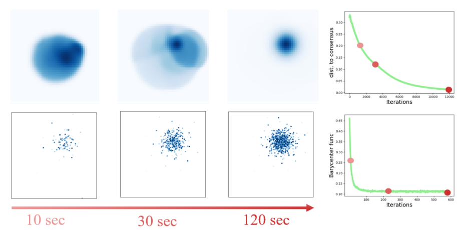

Figure 2: Evolution of the barycenter of 5 Gaussians computed by [18] (Top row) and our algorithm (Bottom) over time. (Right column) distance to consensus (see [18]) and the B_ε functional (with markers at 10, 30, 120 sec).

R1 1. We thank the reviewer for the additional reference, which we will add to the paper. **2.** There are several options for the MINIMIZE routine: for $\mathcal{X}$ finite (e.g., images, such as our experiments on ellipses and k-means), we evaluate the function at all points of $\mathcal{X}$ (in a vectorized way) and select the minimizer by a sorting algorithm. If $\mathcal{X}$ is a continuous domain, we rely on first order methods (e.g. Gradient Descent) applied in parallel to multiple starting points. For instance, in our experiments on Gaussians, we used the python `scipy.optimize` routine as a plugin optimizer.

R2 1. We thank R2 for the reference "Entropic regularization of continuous optimal transport problems". We note that this work studies regularization with *negative entropy of the transport plan* π, whereas in our problem (1) the regularizer is the *Kullback-Leibler* between π and $\alpha \otimes \beta$. These two problems are not equivalent, e.g., if α or β are not absolutely continuous wrt Lebesgue. In our case, existence of solutions is a consequence of [28], which studies DAD problems in *continuous settings*. Indeed, existence of maximizers for (2) is equivalent to existence of solutions to the optimality equation (4), which is a special case of a DAD problem: thus, existence of maximizers for (2) follows from [28, Thm 1]. We cited [28] on line 67 and also discussed this issue in detail in Appendix B.2 (see Cor B.6). **2.** The interpretation of (2) as primal and (1) as dual is indeed the way to derive strong duality, since in this interpretation the qualification conditions hold (see e.g., the proof of Thm. 3.2 in [8]). However, problem (2) can also be seen as the dual of (1) when the involved spaces are endowed with the weak topologies (this requires formulating the Fenchel-Rockafellar duality in locally convex spaces [8, Thm A.1]); this follows the convention used in optimal transport literature. **3.** The work [28] is in infinite dimensional setting (see also the convergence of the Sinkhorn-Knopp algorithm in Appendix B.3.)

R3 1. The exponential slow-down wrt ε reflects recent findings on the Sinkhorn divergence, where similar scaling was observed for, e.g., its sample complexity [21]. However, as also reported in Table 1 above, in practice the slow-down does not seem too severe. In the paper we used $\varepsilon = 0.001$, which typically yields visually good results. **2.** According to Thm 3 the convergence rate depends on the Lipschitz constant of the gradient of the objective function. Since $\nabla \mathsf{B}_\varepsilon$ is a weighted sum of m mappings with same Lipschitz constant and the weights ω_i sum to 1, the Lipschitz constant of $\nabla \mathsf{B}_\varepsilon$ does not depend on m. **3.** Scaling: we computed the barycenter of m distributions with n points each (obtained by randomly displacing and sampling from the 2D distribution in Fig. 1). Table 2 shows the runtimes of our algorithm as m and n vary. We observed that the main bottleneck of our method are the m SINKHORNKNOPP computations at each iteration (e.g. on average $\sim 94\%$ of the total time for $m = 100$ $n = 1000$). We plan to address this by i) parallelizing with respect to the m distributions and ii) adopting the very recent toolbox for Sinkhorn computation `www.kernel-operations.io/geomloss/`, which can yield a $\sim$ 50-100$\times$ speed-up to SINKHORNKNOPP. Such a boost would allow us to consider larger scale settings. **4.** Initialization: a single Dirac Delta randomly sampled in $\mathcal{X}$.