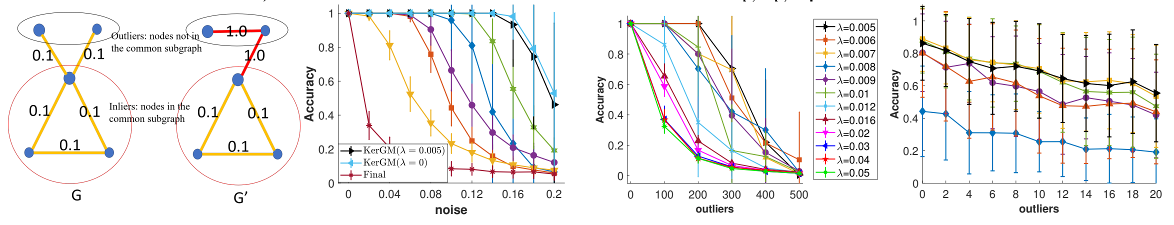
KerGM: Kernelized Graph Matching (Paper ID:1849)

To Reviewer 2

Q1: Connection between the Koopmanns-Beckman QAP (KB-QAP) and the Lawler QAP(L-QAP).
A1: In the literature, it's well-known that KB-QAP can be written in the form of L-QAP, which happens only when edge attributes are scalar and their similarity function is simple multiplication. In our work, we consider the inverse direction. That is, by introducing $\mathcal{H}-$operations, we can write the L-QAP in the KB's form, i.e., the KB alignment between Hilbert arrays, allowing graphs with complex (e.g., vectorial) edge attributes. The KB's alignment form gives us effective convex-concave relaxations, and significant reduction of the time and space complexity of the L-QAP.
Q2: Difference with the work [42].
A2: Previous work [42] replaces the discrete term $\boldsymbol{X}_{ia}\boldsymbol{X}_{jb}$ with a family of continuous functions f_δ to gradually solve the discrete L-QAP, where f_δ relates to kernels. However, in our settings, the edge affinity terms $\boldsymbol{K}_{ij,ab}$ (notation $\boldsymbol{A}_{ij:ab}$ in [42]) are kernel values, which leads to significantly different strategies of solving L-QAP, e.g., how to make relaxations. One important advantage of our work is that our algorithm can scale to dense graphs with thousands of nodes ($n = O(10^3)$), while work [42] cannot because it requires computing the $n^2 \times n^2$ matrix $\boldsymbol{K}$ (notation $\boldsymbol{A}$ in [42]).
Q3: Contributions of this work.
A3: As summarized by Reviewer 3 and 4, we developed **(1).** a unified perspective of two QAPs, **(2).** an efficient entropy-regularized Frank-Wolfe algorithm for optimization, **(3).** the KerGM framework for solving graph matching problems. We agree that kernels are popular in measuring similarities. However, we believe this is the first work of solving the large-scale Lawler's QAP by aligning arrays in RKHS. There exist works that leverage the path-following strategy to solve L-QAP, and they differ with each other in how to relax the original problem. We proposed totally new convex-concave relaxations, and more notably, we made it scalable to large graphs.



(a) (b) (c) (d)

Figure A: (a) an example of (in)outliers; (b) comparing accuracies; (c) sensitivity of λ; (d) comparing accuracies on Pascal dataset.

To Reviewer 3

Q1: (1). Provable runtime of the EnFW subroutine. **(2).** How does the regularizer affect performance?
A1: (1). At each step, the Sinkhorn-Knopp algorithm converges at the linear rate, i.e., $0 < \limsup \|\boldsymbol{X}_{k+1} - \boldsymbol{X}^*\| / \|\boldsymbol{X}_k - \boldsymbol{X}^*\| < 1$, as proved in the Section 4 of the paper "George W.Soules, The rate of convergence of Sinkhorn balancing". The outer iteration convergence rates are shown in Theorem 1 and 2 in our paper. **(2).** We re-conduct the second random graph matching experiments with fixed outliers and edge density, and varying noise. In Fig. A(b), we compare EnFW ($\lambda = 0.005$) and the exact FW ($\lambda = 0$) that uses the Hungarian algorithm during each outer iteration, both of which are under our KerGM formulation. The exact FW performs slightly better than EnFW.
Q2: Compare "KerGM" with "Final" (Zhang et al. 2016) and "Regal" (Heimann et al. 2018)
A2: In Fig. A(b), we show the results of "Final". The standard experimental protocol may be not suitable for "Final" since it doesn't directly solve the Lawler's QAP. For "Regal", we are still trying to fit the code in our settings.
Q3: Definition of the outliers. **A3:** We show the concept of outliers in Fig. A(a), where G and G' are weighted graphs.
Q4: Where does the $\sqrt{\cdot}$ come from in Line 140 in the supplementary material?
A4: Thanks for the careful review. We are sorry that there is a little mistake. Line 140 should be changed into $\Delta_0 \geq (T+1)G_T^*/2 \Rightarrow G_T^* \leq 2\Delta_0/(T+1) \leq 2\Delta_0/\sqrt{T+1}$. The statement in Theorem 1 becomes $G_T^* \leq 2\max\{\Delta_0, \sqrt{L\Delta_0/n}\}/\sqrt{T+1}$.

To Reviewer 4

A1: What is $\boldsymbol{K}^P$? **Q1:** We are sorry that it's a typo. It should be $\boldsymbol{K}^N$, the node attributes affinity matrix.
Q2: Explanation on computational complexity
A2: Existing methods require pre-computing the affinity matrix $\boldsymbol{K} \in \mathbb{R}^{n^2 \times n^2}$, whose both the time and space cost are $O(n^4)$ for dense graphs. In each outer iteration of optimization, the time cost of computing gradient is also $O(n^4)$, because it involves the term $\boldsymbol{K}\text{vec}(\boldsymbol{X})$. Our KerGM doesn't require pre-computing $\boldsymbol{K}$. With the approximate feature map, the space cost is $O(Dn^2)$ and the time cost of computing gradients is $O(Dn^3)$, where $D << n$ (see Section 4.2).
Q3 (experiments): (1). More points for sensitivity of λ. **(2).** Standard deviations of matching results on Section 6.2.
A3: (1). We added more variants of λ in Fig. A(c). **(2).** The results in Section 6.2 are the averaged accuracies of graph matching between many pairs of images. We will add the error bars in the revised paper. Fig. A(d) shows an example.
Q4: How have the parameters in Section 6.3 been selected? **A4:** We tried different combinations of these parameters, and found that the results are not sensitive to them except for the parameter γ, where we select from $\{2, 20, 200, 2000\}$.
A*: Thanks for other comments on organization, writing style, and notations. We will follow them in the revised paper.