A sparse algorithm for dense optimal transport

Bernhard Schmitzer

CEREMADE, Université Paris-Dauphine

Abstract. Discrete optimal transport solvers do not scale well on dense large problems since they do not explicitly exploit the geometric structure of the cost function. In analogy to continuous optimal transport we provide a framework to verify global optimality of a discrete transport plan locally. This allows construction of a new sparse algorithm to solve large dense problems by considering a sequence of sparse problems instead. Any existing discrete solver can be used as internal black-box. The case of noisy squared Euclidean distance is explicitly detailed. We observe a significant reduction of run-time and memory requirements.

1 Introduction and Related Work

Optimal transport is a powerful and popular tool in image analysis, computer vision and statistics (e.g. [14,13,17]). However it is computationally more involved as 'simple' similarity measures such as L^p-distances or the Kullback-Leibler divergence. Consequently there is a necessity for efficient solvers.

Broadly speaking there are two classes of solvers: There are discrete combinatorial algorithms such as the Hungarian method [11], the auction algorithm [4], the network simplex [2] and more (e.g. [9]). They work for (almost) arbitrary cost functions, and are typically numerically robust w.r.t. input data regularity. They do not scale well for large, dense problems however, because the geometric structure of the cost function is not used. Alternatively, there are continuous solvers, based on the elegant theory of the 2-Wasserstein space on $\mathbb{R}^n$ [10,6]. These need not handle the full product space, but work directly with a transport map and so can solve large problems more efficiently. But they only apply to a restricted family of cost functions (most prominently squared Euclidean distance) and they are numerically more subtle (e.g. involving the Jacobian of the transport map), thus requiring some data regularity. The celebrated fluid-dynamics formulation [3] is more flexible but at the cost of introducing a time dimension. Moreover, there is a wide range of approximate methods: cost function thresholding [13], tangent space approximation [17] and entropy smoothing [7] among others.

In [12] and [15] computation time is reduced via multi-scale schemes: first solve a coarse approximate problem and then go to increasingly finer resolutions. However, [12] is limited strictly to the case of squared Euclidean distance and [15] only uses the cost function structure implicitly by keeping the problem sparse by hierarchical consistency checks, requiring low-level adaptions of the algorithm.

So there still is a need for efficient discrete exact solvers, that are more flexible than the W_2-scenario (both in terms of cost function and measure regularity), but which are still able to exploit geometric structure of the cost function.

An important feature of continuous solvers is that under suitable conditions optimality of the transport can be verified by a local criterion: the transport map is the gradient of a convex function. Whereas discrete solvers must check optimality globally (e.g. all dual constraints must be verified).

Contribution and Organization. We briefly recall discrete optimal transport in Sect. 2. Then a framework for the discrete setting is designed to mimic the continuum feature of locally verifying global optimality. Locally here means that we only need to look at a sparse subset of the full product space (Sect. 3). Based on this we propose a new algorithm that globally solves the dense problem via a sequence of sparse problems (Sect. 4). We show explicitly how the algorithm can be applied to the (noisy) squared Euclidean distance on $\mathbb{R}^n$ (Sect. 5).

Key features of the proposed algorithm are: **(a)** It works with any discrete OT solver as internal solver. **(b)** Due to sparsity the iterations are fast. **(c)** It can benefit from smart initialization: when the initial coupling is already close to being optimal, only few iterations are needed. **(d)** Consequently it can be used in a multi-scale scheme, where we obtain good initial guesses for fine scales from solutions on coarse scales. **(e)** This yields a significant decrease in run-time and memory requirements compared to naïve solving of dense problem.

Numerical examples are given in Sect. 6. We report a speed-up of one order of magnitude on state-of-the-art solver software, when simply used as black box and two orders of magnitude for the Hungarian method with smart re-initialization. This was observed both on smooth as well as locally concentrated measures, thus indicating a wide range of practical applicability. Hence, the algorithm can be used when the noisy cost function or irregular marginals cannot be handled by continuous solvers but when naïve combinatorial algorithms are too slow.

2 Background on Optimal Transport

Notation. For a discrete finite set A we write $|A|$ for its cardinality. Denote by $\mathcal{M}(A)$ the space of non-negative measures over A. For a measure $\mu \in \mathcal{M}(A)$ its support is defined by $\operatorname{spt} \mu = \{a \in A \colon \mu(a) > 0\}$. The space of real functions over A is identified with $\mathbb{R}^{|A|}$ where we index the components by elements of A. For a map $f : A \to B$ and a measure $\mu \in \mathcal{M}(A)$ we denote by $f_\sharp \mu \in \mathcal{M}(B)$ the *push-forward* of μ given by $f_\sharp \mu(\sigma) = \mu(f^{-1}(\sigma))$ for $\sigma \subset B$.

Discrete Optimal Transport. For two discrete finite sets X, Y and two non-negative measures $\mu \in \mathcal{M}(X)$, $\nu \in \mathcal{M}(Y)$ with equal total mass $\mu(X) = \nu(Y)$ the set of couplings is given by

$$\Pi(\mu,\nu) = \{\pi \in \mathcal{M}(X \times Y) \colon \pi(\{x\} \times Y) = \mu(x),\, \pi(X \times \{y\}) = \nu(y)\ \forall\, x, y\} \tag{2.1}$$

For a cost function $c : X \times Y \to \mathbb{R} \cup \{\infty\}$ the optimal transport problem consists of finding the coupling with minimal total transport cost:

$$\min_{\pi \in \Pi(\mu,\nu)} C(\pi) \qquad \text{with} \qquad C(\pi) = \sum_{(x,y) \in X \times Y} c(x,y)\, \pi(x,y) \tag{2.2}$$

The problem is called feasible if its optimal value is finite. We call (2.2) the *dense* or *full problem*. For some $N \subset X \times Y$ we also consider problem (2.2) subject to the additional constraint $\operatorname{spt} \pi \subset N$, which we call the problem *restricted to* N. We call N a *neighbourhood*. And we will call π a local optimizer w.r.t. N if it solves the corresponding restricted problem.

The dual problem to (2.2) is given by

$$\max_{(\alpha,\beta)\in(\mathbb{R}^{|X|},\mathbb{R}^{|Y|})} \sum_{x\in X} \alpha(x)\,\mu(x) + \sum_{y\in Y} \beta(y)\,\nu(y) \tag{2.3a}$$

$$\text{subject to} \quad \alpha(x) + \beta(y) \leq c(x,y) \quad \text{for all} \quad (x,y) \in X \times Y\,. \tag{2.3b}$$

The relation between any primal and dual optimizers π and (α,β) of the same transport problem is

$$\pi(x,y) > 0 \qquad \Rightarrow \qquad \alpha(x) + \beta(y) = c(x,y)\,. \tag{2.4}$$

Restricting the primal problem to N corresponds to only enforcing the dual constraints (2.3b) on N. Analogously we speak of local dual optimizers (α,β) w.r.t N. If $(\pi,(\alpha,\beta))$ are local primal and dual optimizers and (α,β) satisfy (2.3b) on $X \times Y$, then one has found optimizers for the full problem.

One goal of this paper is to find suitable small subsets N such that the local optimizers $(\pi,(\alpha,\beta))$ w.r.t. N are also optimal for the full problem.

3 Optimal Transport and Short-Cuts

An Example for Intuition. Let μ,ν be absolutely continuous measures on $\mathbb{R}^n$ with compact convex support, let $c(x,y) = \|x-y\|^2$. Then we know that there is an optimal transport map which is the gradient of a convex function [16]. Let T be any transport map, $T_\sharp\mu = \nu$, with induced coupling $\pi = (\mathrm{id},T)_\sharp\mu$. For simplicity let T be a homeomorphism. We want to verify optimality of T.

Let $\{U_i\}_i$ be an open covering of $\operatorname{spt}\mu$, then $\{V_i\}_i$ with $V_i = T(U_i)$ is an open covering of $\operatorname{spt}\nu$. Let $\mu|_{U_i}$, $\nu|_{V_i}$ be the restrictions of the measure μ to U_i and ν to V_i. Then T is a transport map between $\mu|_{U_i}$ and $\nu|_{V_i}$ for all i. If T is optimal for each restricted problem on $U_i \times V_i$ then optimality for the whole problem follows: when we know that T is the gradient of a convex function on each U_i, by convexity of $\operatorname{spt}\mu$ it follows that T is the gradient of a convex function on $\operatorname{spt}\mu$ and thus is the optimal transport map. Since the patches U_i can be made arbitrarily small, optimality of a coupling π can be verified on an arbitrary small open environment of $\operatorname{spt}\pi$ on $(\mathbb{R}^n)^2$. This is illustrated in Fig. 1 (left).

The *Monge property* [5] is a simple discrete analogy in one dimension. In this paper we strive to find a discrete equivalent for higher-dimensional problems. We will return to this discussion for a brief comparison in Sect. 5. Now we introduce short-cuts, a tool to temporarily remove constraints from the dual problem.

Definition 1 (Short-Cut). *For a neighbourhood $N \subset X \times Y$ and a coupling π with $\operatorname{spt}\pi \subset N$ let $((x_1,y_1),\ldots,(x_n,y_n))$ be an ordered tuple of pairs in $\operatorname{spt}\pi$.*

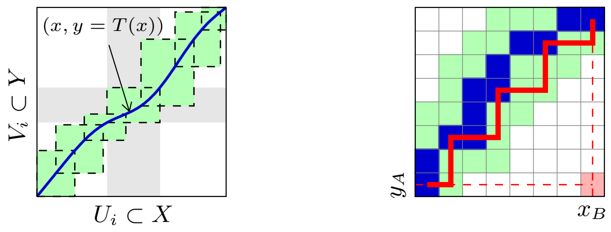

Fig. 1. *Left:* In the continuous case it suffices to check whether T is optimal on each of the sets $U_i \times V_i$. Global optimality then follows. *Right:* As a discrete analogy we introduce the concept of *short-cuts*. The dual constraint at (x_B, y_A) is implied by a sequence (solid red line) of points in $\operatorname{spt}\pi$ (blue) such that the 'jumps' lie in N (green).

We say $((x_1, y_1), \ldots, (x_n, y_n))$ *is a* short-cut between $(x_B, y_A) \in (X \times Y) \setminus N$ *if* $x_B = x_n$*,* $y_A = y_1$*,* $(x_{i+1}, y_i) \in N$ *for* $i = 1, \ldots, n-1$ *and*

$$c(x_B, y_A) = c(x_n, y_1) \geq c(x_2, y_1) + \sum_{i=2}^{n-1} [c(x_{i+1}, y_i) - c(x_i, y_i)] \,. \tag{3.1}$$

Proposition 1. *For a set* $N \subset X \times Y$ *let* $(\pi, (\alpha, \beta))$ *be a pair of local primal and dual optimizers. Assume for a pair* $(x_B, y_A) \notin N$ *there exists a short-cut within* N*. Then the dual constraint* (2.3b) *corresponding to* (x_B, y_A) *is satisfied.*

Proof. Let $((x_1, y_1), \ldots, (x_n, y_n))$ be a short-cut. From (2.4) and (2.3b) restricted to N we find $\alpha(x_{i+1}) - \alpha(x_i) \leq c(x_{i+1}, y_i) - c(x_i, y_i)$ for $i = 1, \ldots, n-1$ and by summing up over i we obtain

$$\alpha(x_B) + \beta(y_A) = \alpha(x_n) + \beta(y_1) \leq c(x_2, y_1) + \sum_{i=2}^{n-1} [c(x_{i+1}, y_i) - c(x_i, y_i)] \,.$$

Validity of the dual constraint corresponding to (x_B, y_A) follows from (3.1). □

The concept of short-cuts is illustrated in Fig. 1 (right). Dual constraints for which a short-cut exists need no longer be checked. So is there a clever way to choose a small set N such that there is a short-cut for every $(x_B, y_A) \notin N$? But explicitly checking existence of short-cuts for each pair is far too expensive. We now introduce a simple sufficient condition for existence.

Definition 2 (Shielding Condition). *For a coupling* π *let* $y_A \in Y$*,* $(x, y) \in \operatorname{spt}\pi$ *and* $x_B \in X$*. We say* (x, y) *shields* y_A *from* x_B *when*

$$c(x_B, y_A) - c(x_B, y) > c(x, y_A) - c(x, y) \,. \tag{3.2}$$

The shielding condition states that $\{(x, y_A), (x_B, y)\}$ is ('strictly') c-cyclically monotone [16]. It implies that suitable n-tuples in $\operatorname{spt}\pi$ are in fact short-cuts.

Proposition 2. *For a given coupling π let $((x_1, y_1), \ldots, (x_n, y_n))$ be an ordered tuple in $\operatorname{spt} \pi$. If $(x_{i+1}, y_i) \in N$ for $i = 1, \ldots, n-1$ and (x_{i+1}, y_{i+1}) shields y_i from x_n for $i = 1, \ldots, n-2$ then the tuple is a short-cut for (x_n, y_1).*

Proof. We need to show that (3.1) holds. For $i = 1, \ldots, n-2$ we have from (3.2)

$$c(x_n, y_i) - c(x_n, y_{i+1}) > c(x_{i+1}, y_i) - c(x_{i+1}, y_{i+1}) \,.$$

Summing up yields $c(x_n, y_1) > \sum_{i=2}^{n-1} [c(x_i, y_{i-1}) - c(x_i, y_i)] + c(x_n, y_{n-1})$. □

We now introduce a sufficient condition for a set N such that short-cuts exist for all $(x_B, y_A) \notin N$ and an algorithm for construction.

Definition 3 (Shielding Neighbourhood). *For a given coupling π we say that a neighbourhood $N \subset X \times Y$, $N \supset \operatorname{spt} \pi$ is* shielding *if for every $y_A \in Y$ and any $x_B \in X$ at least one of the following is true:*

(i) $(x_B, y_A) \in N$.
(ii) *There exists some $(x, y) \in \operatorname{spt} \pi$ with $(x, y_A) \in N$ such that (x, y) shields y_A from x_B.*

Algorithm 1 *For a neighbourhood N let $(\pi, (\alpha, \beta))$ be the corresponding local optimizers. Assume N is shielding for π. We will construct a short-cut for a given pair $(x_B, y_A) \notin N$. Set $n \leftarrow 1$, $y_1 \leftarrow y_A$ and choose some x_1 such that $(x_1, y_1) \in \operatorname{spt} \pi$. Then iterate:*

```
while $(x_B, y_n) \notin N$:
    find $(x_{n+1}, y_{n+1}) \in \operatorname{spt} \pi$ with $(x_{n+1}, y_n) \in N$ such that
        $(x_{n+1}, y_{n+1})$ shields $y_n$ from $x_B$; $n \leftarrow n+1$
end while
$x_{n+1} \leftarrow x_B$; pick some $y_{n+1}$ such that $(x_{n+1}, y_{n+1}) \in \operatorname{spt} \pi$; $n \leftarrow n+1$
```

Proposition 3 (Existence of Short-Cuts). *Under the stated requirements Algorithm 1 terminates and produces a valid short-cut for any pair $(x_B, y_A) \notin N$.*

Proof. By virtue of Definition 3 one always finds either $(x_B, y_n) \in N$ or there exists a suitable shielding (x_{n+1}, y_{n+1}). Since the number of elements in $\operatorname{spt} \pi$ is finite, either the iteration must eventually terminate, or a cycle occurs. The existence of cycles along which the shielding condition holds is ruled out by (2.3b) and (2.4). Consequently the algorithm terminates. Then Proposition 2 provides that $((x_1, y_1), \ldots, (x_n, y_n))$ is a short-cut for (x_B, y_A). □

Remark 1. Note that the strict inequality in (3.2) is merely required to guarantee termination of Algorithm 1. Proposition 2 already follows from $\geq$ in (3.2).

Running Algorithm 1 is immensely more expensive than checking the corresponding dual constraint. We rely on Proposition 3 which directly implies existence of short-cuts and thus validity of constraints outside of N. We summarize:

Corollary 1 (Global Optimality from Local Optimality). *Let $(\pi, (\alpha, \beta))$ be local optimizers w.r.t. a neighbourhood N and let N be shielding for π. Then $(\pi, (\alpha, \beta))$ are optimizers of the dense problem.*

Remark 2. A primal local optimizer π w.r.t. N suffices for Corollary 1 to hold as shielding of N only depends on π. Results hold for any matching local dual optimizers (α, β). Explicitly knowing dual variables allows for verifying local optimality by checking the dual constraints.

4 A Sparse Algorithm.

Corollary 1 can be used to construct an efficient sparse algorithm for large OT problems. The main ingredients of the algorithm are two maps:

(i) $F : N \mapsto \pi$ such that $F(N)$ is locally optimal w.r.t. N. When N is sparse, any discrete OT solver can quickly provide an answer. Given Remark 2 we see that also purely primal solvers suffice.
(ii) $G : \pi \mapsto N$ such that $G(\pi)$ is shielding for π. It is important for efficiency that $G(\pi)$ is sparse. To design such a map one must use the geometric structure of the cost function. In Sect. 5 we discuss the squared Euclidean distance.

Corollary 1 entails a ‘chicken and egg’-problem: For a given N_1 let $\pi_1 = F(N_1)$. But if π_1 is not globally optimal, then N_1 cannot be shielding w.r.t. π_1. Conversely, for some π_1 let $N_2 = G(\pi_1)$, but π_1 will only be locally optimal w.r.t. N_2 iff it is globally optimal. To find a configuration (N, π) such that both criteria are satisfied simultaneously, one can iterate both maps.

Algorithm 2 *For an initial N_1 set $k = 1$ and run:*

```
do:
    $\pi_{k+1} \leftarrow F(N_k)$; $N_{k+1} \leftarrow G(\pi_{k+1})$; $k \leftarrow k+1$
until $C(\pi_k) = C(\pi_{k-1})$
```

Proposition 4. *For a feasible initial N_1 Algorithm 2 terminates after a finite number of steps with a globally optimal π_k.*

Proof. For $k > 1$ have $N_k = G(\pi_k)$ and $\pi_{k+1} = F(N_k)$. So $\operatorname{spt} \pi_k \subset N_k$ and therefore π_k is feasible when computing the optimal π_{k+1}, restricted to N_k. It follows $C(\pi_{k+1}) \leq C(\pi_k)$. One can see that after a finite number of iterations one must find $C(\pi_k) = C(\pi_{k+1})$. Then π_k is optimal w.r.t. N_k and by construction N_k is a shielding w.r.t. π_k. Therefore π_k and π_{k+1} are globally optimal.

The usefulness of this algorithm will be demonstrated numerically in Sect. 6. Important advantageous properties are:

(i) The solver F must only be applied to sparse problems, thus calling F will be quite fast, if a suitable function G is known.
(ii) In fact we will demonstrate in next section how G can be designed for squared Euclidean distance and extensions thereof, to quickly provide sparse sets N.
(iii) When a good initial coupling π_1 is known, the algorithm will only need few iterations. This is ideal for working in a multi-scale scheme, where an initial guess for π_1 or N_1 can be generated from a coarser version of the problem.

5 Squared Euclidean Distance

The Shielding Condition on $\mathbb{R}^n$. So far everything has been formulated for general cost functions. To run Algorithm 2 we need a function G that quickly generates a sparse shielding neighbourhood N for a given coupling π. For this we must exploit the particular geometric structure of the cost function. We now explicitly show how to do this for the squared Euclidean distance and noise.

For now assume $X, Y \subset \mathbb{R}^n$ and $c(x,y) = \|x-y\|^2$. Then the shielding condition (3.2) for some triple $y_A \in Y$, $(x,y) \in \operatorname{spt}\pi$, $x_B \in X$ is equivalent to

$$\langle x_B - x, y - y_A \rangle > 0\,. \tag{5.1}$$

This equation has a simple geometric interpretation: consider the hyperplane through x, normal to $y - y_A$. Then x_B must lie on the side facing in direction $y - y_A$. So (x,y) shields y_A from all potential x_B beyond this hyperplane. This provides us with a recipe for constructing shielding neighbourhoods:

```
$N \leftarrow \operatorname{spt}\pi$
for every $y_A \in Y$:
    find a set $\{(x_i,y_i)\}_i \subset \operatorname{spt}\pi$ such that the hyperplanes with
        normals $y_i - y_A$ through $x_i$ form a small polytope $P \subset \mathbb{R}^n$.
    add the pairs $(x_i, y_A)$ to $N$ for all $i$ // (step-i)
    for all $x' \in X \cap P$: add $(x', y_1)$ to $N$. // (step-ii)
```

After (step-i) y_A is shielded from all x' outside of P by at least one (x_i, y_i), after (step-ii) from all $x' \in X \cap P$. Now we discuss finding suitable polytopes P.

Regular Grids on $\mathbb{R}^2$. Assume X and Y are regular orthogonal grids. For simplicity assume $n = 2$, higher dimensions work analogously. From a coupling π one can extract a map $T_\pi : Y \to X$ with $x = T_\pi(y) \Rightarrow \pi(x,y) > 0$.

For any $y_A \in Y$ let $\{y_1, y_2, y_3, y_4\}$ be the 4-neighbourhood of y_A on the grid Y and let $x_i = T_\pi(y_i)$ for $i = 1, \ldots, 4$. The normals of the faces of the resulting polytope P are therefore $\{(1,0), (0,1), (-1,0), (0,-1)\}$ and P is a bounded rectangle, aligned with the grid. So during (step-i) of the construction of N for each $y_A \in Y$ we add four elements (x_i, y_A), $i = 1, \ldots, 4$ to N. During (step-ii) we add (x', y_A) for all $x' \in X \cap P$. Both the 4-neighbourhood of y_A and elements of the interior of P can be accessed in $\mathcal{O}(1)$ by using the grid structure on X and Y.

Provided some auxiliary data structures the scheme can be extended to more general point clouds, this is however beyond the scope of this paper.

Complexity. The critical components for the complexity of Algorithm 2 are: complexity characteristics of the internal solver F, sizes of neighbourhoods $\{N_k\}_k$ and the number of outer iterations. The latter two depend on the initial choice of N_1. A rigorous analysis of this dependency is beyond the scope of this paper. But beyond the empirically observed speed-up (Sect. 6) we can provide some intuition on the algorithm's behaviour: Given a spatially regular π (nearby x assign mass to nearby y), the points x_i are close and therefore P will have a small area (in particular $\mathcal{O}(1)$ w.r.t. $|X|$ and $|Y|$) and only few elements are added during (step-ii). So the resulting N will indeed be sparse, $|N| = \mathcal{O}(|Y|)$. This scaling will be confirmed numerically for the multi-scale initialization scheme that we chose (Sect. 6). We will even find $\sum_k |N_k| = \mathcal{O}(|X|)$, i.e. the sum of the neighbourhood-sizes over all iterations of Algorithm 2, still scales better than for the dense problem where $N = X \times Y$. Since the internal solver complexity is super-linear in $|N_k|$, this already implies a gain in performance.

Beyond Squared Euclidean Distance. Now consider a more general cost function:

$$c(x,y) = \|x-y\|^2 + \hat{c}(x,y) \qquad \text{where we assume} \qquad 0 \leq \hat{c}(x,y) \leq \delta \tag{5.2}$$

Such a cost function can describe a matching problem where we do not only consider geometric proximity of points x and y, but also additional descriptors attached to x and y (e.g. SIFT descriptors), whose comparison cost is given by $\hat{c}(x,y)$. Albeit useful from a modelling perspective, adding a term $\hat{c}$ to the cost function destroys the 2-Wasserstein space structure of the problem and thus solvers based on the polar factorization theorem can no longer be applied.

It is possible however to extended our method to this scenario. Consider (3.2) for this cost function. We find it is equivalent to

$$\langle x_B - x, y - y_A \rangle > \frac{1}{2} \left(\hat{c}(x,y_A) - \hat{c}(x,y) - \hat{c}(x_B,y_A) + \hat{c}(x_B,y) \right) . \tag{5.3}$$

By using (5.2) we find the simpler sufficient condition $\langle x_B - x, y - y_A \rangle > \delta$. So to construct a shielding neighbourhood for such a cost function we can still apply the recipe outlined above, but all faces of P must be shifted outwards by δ to account for the fluctuations in $\hat{c}$. For each x_B within this enlarged polytope we can test via (5.3) whether (x_B, y_A) must be included into N.

Application to the Continuous Problem. We now return to the discussion in Sect. 3 and relate our new results to the continuous formulation. Given a transport map T, locally optimal on all the patches $U_i \times V_i$, let the tuple points $(x_1 = x_A, \ldots, x_n = x_B)$, $x_i \in \operatorname{spt} \mu$, be taken from a straight line between x_A and x_B in monotone order and picked fine enough such that every two successive points x_i, x_{i+1} lie in the same patch U_i. Let $y_i = T(x_i)$. It then follows that (x_{i+1}, y_{i+1}) is shielding y_i from x_n for $i = 1, \ldots, n-2$ (see (5.1) and Remark 1) and consequently the tuple $((x_1,y_1), \ldots, (x_n,y_n))$ is a short-cut for $(x_B, y_A = y_1)$. Therefore the transport map T is globally optimal.

We see that the shielding condition follows from local optimality along straight lines, which explains why local optimality is still sufficient in 1-d discrete problems. In discrete higher-dimensional problems we cannot always jump along straight lines between grid points and even small deviations may break the shielding condition. Thus we must explicitly keep track of π in Sect. 3.

6 Numerical Experiments

Algorithms and Adaptive Initialization. We test our algorithm on four discrete solvers: Our own implementation of the Hungarian method [11], the network

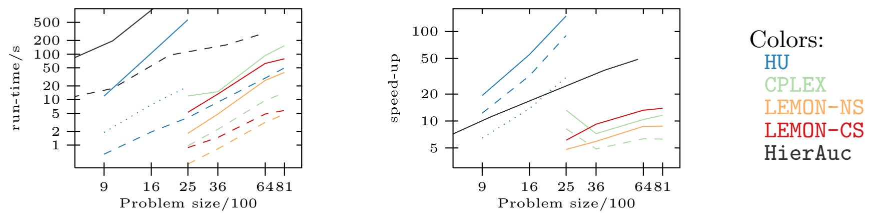

Fig. 2. Run-times and speed-up. Problem size refers to $|X| = |Y|$. Left: Run-times of naïve dense algorithms (solid lines) and Algorithm 2 with a multi-scale scheme (dashed lines). For `HU`: `HU-STD` dotted, `HU-INIT` dashed. Right: Relative speed-up for each algorithm. `HU-STD` dotted, `HU-INIT` solid. Dashed lines: results with noisy cost function (Sect. 5) for `HU` and `CPLEX`. We observe a huge speed-up for `HU`, which is even larger with adaptive initialization. But also the high-performance external solvers can be accelerated by about one order of magnitude for large problems. The speed-up tends to increase with problem size. `HierAuc` gives the run-times and speed-up reported in [15] for the scenario `grid`. The auction algorithm does not seem to perform well on this problem class and both the run-time and speed-up obtained by `HU-INIT` are better.

simplex [2] implementation of CPLEX [1] and the network simplex and cost scaling [9] implementations of the LEMON library [8]. In the following we refer to these four algorithms by the short-hands `HU`, `CPLEX`, `LEMON-NS` and `LEMON-CS`.

While we expect `CPLEX` and `LEMON-`$*$ to be faster than `HU`, they can only be used as black boxes, whereas we can manipulate our own implementation in more detail. In particular we can choose the initialization. This can be exploited by Algorithm 2: instead of solving each sparse problem $(\pi_{k+1}, (\alpha_{k+1}, \beta_{k+1})) \leftarrow F(N_k)$ from scratch, we initialize the algorithm using the previous optimizers $(\pi_k, (\alpha_k, \beta_k))$ as follows: We start with $\beta_{k+1,\text{init}} = \beta_k$ and $\alpha_{k+1,\text{init}}(x) = \alpha_k$ and then reduce all $\alpha_{k+1,\text{init}}$ entries where dual constraints on N_k are violated. We set $\pi_{k+1,\text{init}} = \pi_k$ and set all X-rows to zero where $\alpha_{k+1,\text{init}}$ has been reduced. When only few constraints were violated, we start with an 'almost feasible' coupling.

Test Data. As test data we used measures on regular grids in $\mathbb{R}^2$ with full support. Assuming full support is common for continuum solvers (e.g. [10]) and can be ensured by adding a small constant measure. With the exception of `LEMON-CS` we observed that the discrete solvers could handle very small constants and thus distortion of the problem was negligible. We test grid sizes between 30×30 and 90×90, so the cardinalities of X and Y range between 900 and 8100 and the dimensions of the full coupling-spaces between $8.1 \cdot 10^5$ and $6.6 \cdot 10^7$. The tested measures contained smoothly varying densities, strong Dirac-like local concentrations and sharp discontinuities, thus posing challenging problems (see Fig. 4 (right)) and representing a wide range of potential applications.

Multi-scale Solving. The purpose of Algorithm 2 is to accelerate solving large problems by obtaining a smart initial guess for the optimal coupling and then quickly solving a sequence of sparse problems, instead of trying to solve the

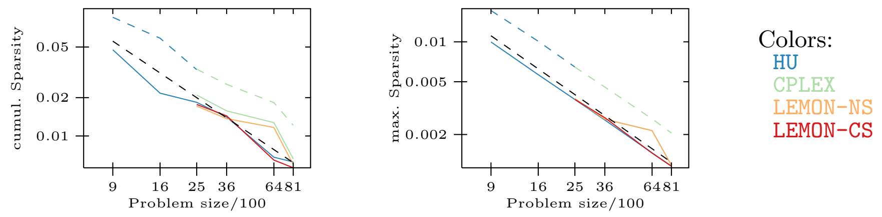

Fig. 3. Solid lines: squared Euclidean distance, dashed lines: noisy cost function. *Left:* Cumulative sparsity. The sum of all $|N_k|/(|X| \cdot |Y|)$ over iterations of Algorithm 2. *Right:* The maximum of $|N_k|/(|X| \cdot |Y|)$ during iterations. The plots illustrate that the neighbourhood construction (Sect. 5) works as intended. Comparison with $\mathcal{O}(|X|^{-1})$ (black dashed line) shows that the number of neighbours per element is $\mathcal{O}(1)$.

dense problem directly. Similar to [12,15] we approximate the original problem by a sequence of successively coarser problems. We use a hierarchical quad-tree clustering over the grids X and Y. At any scale μ and ν are approximated by the masses of the clusters and c by the distance between the cluster centers. Then solve the problem from coarse to fine: each time we use the support of the optimal coupling as initialization for N_1 at the subsequent finer scale.

Run-times. We compare the run-times of the naïve algorithms with using them as internal solvers in Algorithm 2 combined with the multi-scale scheme. For the multi-scale timing we sum the times it takes to solve all levels, from coarse to fine. The observed run-times and the speed-up are illustrated in Fig. 2. Reported run-times were obtained on a single core of an Intel Core i5 processor at 3.2 GHz.

All solvers are sped up significantly, the ratio increasing with problem size. As expected `HU` is much slower than the other algorithms. But it allows to demonstrate the benefit of adaptive re-initialization over solving from scratch. Currently adaptive re-initialization for the external implementations is not available. But even by using them as black boxes one gains about one order of magnitude.

Sparsity and Number of Iterations. The demonstrated speed-up relies on the sparsity of N_k which also reduces the memory requirements of the algorithms. Our numerical findings are presented in Fig. 3. The ratio $|N_k|/|X \times Y|$ is consistently decreasing for all applied solvers and we observe $|N_k| = \mathcal{O}(|X| = |Y|)$, i.e. the number of neighbours per element in X or Y, $|N_k|/|X|$, is constant w.r.t. problem size. Even the sum of all $|N_k|$ over the iterations of Algorithm 2 is $\mathcal{O}(|Y|)$. On the test data the median number of iterations per scale with the multi-scale scheme was 4, the 95% quantile was 7 iterations, thus numerically confirming the complexity discussion in Sect. 5 and point (iii) in Sect. 4.

Fig. 4 gives an impression of the structure of N_k during the execution of Algorithm 2. We see that N_k locally adapts to the regularity of the assignment: in regular areas only very few elements in N_k are needed per element of $y \in Y$. In irregular regions the size of the neighbourhood increases, but this is only a local effect. The regular regions are not affected by this.

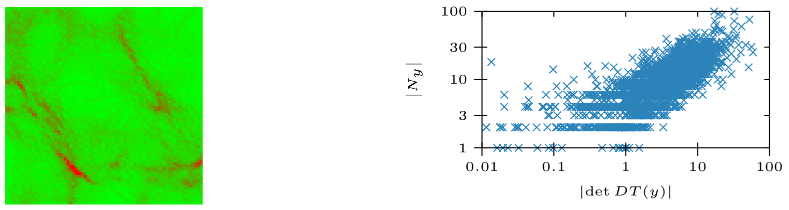

Fig. 4. Let $N_y = N_k \cap (X \times \{y\})$. *Left:* heat-map of $|N_y|$ over Y after one iteration of Algorithm 2 ($k = 2$). Green indicates $|N_y| = 1$, plain red $|N_y|/|X| \geq 1\%$. *Right:* Scatter-plot of $|N_y|$ vs. the Jacobian determinant of the average assignment map $T : Y \to X$ ($T(y)$ is averaged over X w.r.t. $\pi(y, \cdot)$). $|\det DT(y)|$ varies over 4 orders of magnitude, including both strong compression and expansion. We see that the sparsity adapts locally to the spatial regularity of the assignment. In non-expanding regions only few neighbours per element are necessary.

Noise. We also briefly studied the application of Algorithm 2 to 'noisy' Euclidean cost functions (Sect. 5). Uniform noise sampled from $[0, 5]$ was added to the clean cost function (the distance of neighbouring grid points is 1). This naturally increased the cardinality of the neighbourhoods and consequently reduced the speed-up by about a factor 0.6 (see Fig. 2). We still observe $N_k = \mathcal{O}(|X|)$ however (Fig. 3) and consequently observe better speed-up ratios at larger grid sizes.

Comparison with [15]. Fig. 2 compares the speed of the hierarchical auction algorithm (`HierAuc`) and our new method. `HierAuc` is more flexible in terms of cost functions. However, in Algorithm 2 validity of constraints outside of N is directly implied, without the need for hierarchical consistency checks. Therefore it is not restricted to the auction algorithm as internal solver and no adaptions need to be made within the solver. So it can be used with modern high-performance software, to obtain lower run-times. Also, `HU-INIT` yields better speed-up ratios.

7 Conclusion

Dense optimal transport problems are omnipresent. But there is a lack of efficient discrete solvers, exploiting the structure of the cost function.

Our paper provides a means of verifying global optimality of a coupling by only looking at a sparse subset of the full product space. This can be seen as discrete equivalent for well-known continuum results. We showed how to efficiently construct such sets for the squared Euclidean distance and noise. Based thereon we proposed an algorithm that solves dense problems via a sequence of sparse problems. This algorithm can be combined with coarse-to-fine multi-scale solution approaches. We demonstrated numerically the efficiency of the scheme in terms of run-time and sparsity and gave some intuition for the complexity behaviour. Our scheme thus allows the application of discrete solvers to larger problems, where continuum solvers may not be applicable either, due to noisy costs or irregular marginals with strongly fluctuating densities.

Future work will comprise a more detailed complexity analysis, adaptive reinitialization for other algorithms and studying of the shielding condition for other types of cost functions and manifolds.

Acknowledgements. The author gratefully acknowledges support by the German Science Foundation (DFG), grant GRK 1653 and a public grant overseen by the French National Research Agency (ANR) as part of the 'Investissements d'avenir', program-reference ANR-10-LABX-0098.

References

1. CPLEX. http://www.ilog.com.
2. R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., 1993.
3. J.-D. Benamou and Y. Brenier. A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem. *Numerische Mathematik*, 84(3):375–393, 2000.
4. D. P. Bertsekas. A distributed algorithm for the assignment problem. Technical report, Lab. for Information and Decision Systems Report, MIT, May 1979.
5. R. E. Burkhard, B. Klinz, and R. Rudolf. Perspectives of Monge properties in optimization. *Discr. Appl. Math.*, 70(2):95–161, 1996.
6. G. Carlier, A. Galichon, and F. Santambrogio. From Knothe's transport to Brenier's map and a continuation method for optimal transport. *SIAM J. Math. Anal.*, 41:2554–2576, 2010.
7. M. Cuturi. Sinkhorn distances: Lightspeed computation of optimal transportation distances. In *Advances in Neural Information Processing Systems 26 (NIPS)*, pages 2292–2300, 2013. http://arxiv.org/abs/1306.0895.
8. B. Dezsőa, A. Jüttnerb, and P. Kovácsa. LEMON – an open source C++ graph template library. In *Workshop on Generative Technologies (WGT) 2010*, volume 264 of *Electronic Notes in Theoretical Computer Science*, pages 23–45, 2011.
9. A. V. Goldberg and R. E. Tarjan. Finding minimum-cost circulations by successive approximation. *Math. Oper. Res.*, 15(3):430–466, 1990.
10. S. Haker, L. Zhu, A. Tannenbaum, and S. Angenent. Optimal mass transport for registration and warping. *Int. J. Comp. Vision*, 60:225–240, December 2004.
11. H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research Logistics*, 2:83–97, 1955.
12. Q. Mérigot. A multiscale approach to optimal transport. *Computer Graphics Forum*, 30(5):1583–1592, 2011.
13. O. Pele and W. Werman. Fast and robust Earth Mover's Distances. In *International Conference on Computer Vision (ICCV 2009)*, 2009.
14. Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *Int. J. Comp. Vision*, 40(2):99–121, 2000.
15. B. Schmitzer and C. Schnörr. A hierarchical approach to optimal transport. In *Scale Space and Variational Methods (SSVM 2013)*, pages 452–464, 2013.
16. C. Villani. *Optimal Transport: Old and New*, volume 338 of *Grundlehren der mathematischen Wissenschaften*. Springer, 2009.
17. W. Wang, D. Slepčev, S. Basu, J. A. Ozolek, and G. K. Rohde. A linear optimal transportation framework for quantifying and visualizing variations in sets of images. *Int. J. Comp. Vision*, 101:254–269, 2012.