An Asymptotically Optimal Algorithm for Pickup and Delivery Problems

Kyle Treleaven, Marco Pavone, Emilio Frazzoli

***Abstract*— Pickup and delivery problems (PDPs), in which objects or people have to be transported between specific locations, are among the most common combinatorial problems in real-world operations. One particular PDP is the Stacker Crane problem (SCP), where each commodity/customer is associated with a pickup location and a delivery location, and the objective is to find a minimum-length tour visiting all locations with the constraint that each pickup location and its associated delivery location are visited in consecutive order. The SCP is a route optimization problem behind several transportation systems, e.g., Transportation-On-Demand (TOD) systems. The SCP is NP-Hard and the best know approximation algorithm only provides a 9/5 approximation ratio. We present an algorithm for the *stochastic* SCP which: (i) is asymptotically optimal, i.e., it produces a solution approaching the optimal one as the number of pickups/deliveries goes to infinity; and (ii) has computational complexity $O(n^{2+\epsilon})$, where n is the number of pickup/delivery pairs and ϵ is an arbitrarily small positive constant. Our results leverage a novel connection between the Euclidean Bipartite Matching Problem and the theory of random permutations.**

I. Introduction

Pickup and delivery problems (PDPs) constitute an important class of vehicle routing problems in which objects or people have to be transported between locations in a physical environment. These problems arise in many contexts such as logistics, transportation-on-demand systems, and robotics among others. Broadly speaking, PDPs can be divided into three classes [1]: 1) Many-to-many PDPs, characterized by several origins and destinations for each commodity/customer; 2) one-to-many-to-one PDPs, where commodities are initially available at a depot and are destined to customers' sites, and commodities available at customers' sites are destined to the depot (this is the typical case for the collection of empty cans and bottles); and 3) one-to-one PDPs, where each commodity/customer has a given origin and a given destination.

When one adds capacity constraints to transportation vehicles and disallows transshipments, the one-to-one PDP is commonly referred to as the Stacker Crane Problem (SCP). SCPs represent the route optimization problem behind several transportation systems, including delivery truck networks and transportation-on-demand systems, where users formulate requests for transportation from a pickup point to a delivery point [2], [3]. While in general the set of requests to be answered is not known a-priori, random models are often used to describe the behavior of the underlying processes generating requests. Even though the SCP arises in several contexts, current algorithms for its solution are either of exponential complexity or come with quite poor guarantees on their performance. The main contribution of this paper is to develop a *near*-optimal, polynomial-time algorithm for the SCP where origin/destination pairs are (stochastically) dispersed within an Euclidean environment.

Literature overview. We present the main results for the SCP; results about its many generalizations can be found in [1]. The SCP, being a generalization of the Traveling Salesman Problem, is NP-Hard [4]. The problem is NP-Hard even on trees, since the Steiner Tree Problem can be reduced to it [5]. In [5], the authors present several approximation algorithms with a worst-case performance ratio ranging from 1.5 to around 1.21. The 1.5 worst-case algorithm, based on a Steiner tree approximation, runs in linear time. Recently, one of the polynomial-time algorithms presented in [5] has been shown to provide an optimal solution on almost all inputs (with a 4/3-approximation in the worst case) [6]. Even though the problem is NP-hard on general trees, the problem is in P on paths [13]. For general graphs, the best approximation ratio is 9/5 and is achieved by an algorithm in [7]. Finally, an average case analysis of the SCP on trees has been examined for the special case of caterpillars as underlying graphs [8].

Contributions. In this paper, we embed the SCP within a probability framework where origin/destination pairs are identically and independently distributed random variables within an Euclidean environment. Our random model is general in the sense that we consider potentially non-uniform distributions of points, with an emphasis on the case that the distribution of pickup sites is distinct from that of delivery sites; furthermore, the graph induced by the origin/destination pairs does not have any specific restrictions. We refer to this version of the SCP as the *stochastic* SCP. The key contribution of this paper is to present the SPLICE algorithm, a polynomial-time algorithm for the stochastic SCP, which is asymptotically optimal almost surely; that is, except on a set of instances of measure zero, the cost of the tour produced by this algorithm approaches the optimal cost as the number n of origin/destination pairs goes to infinity. In practice, convergence is very rapid and SPLICE computes solutions for the SCP within 5% of the optimal cost for a number of pickup/delivry pairs as low as 100. The SPLICE algorithm has complexity of the order $O(n^{2+\epsilon})$ (for

Kyle Treleaven and Emilio Frazzoli are with the Laboratory for Information and Decision Systems, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139 {ktreleav, frazzoli}@mit.edu.

Marco Pavone is with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109 marco.pavone@jpl.nasa.gov.

This research was supported in part by the Future Urban Mobility project of the Singapore-MIT Alliance for Research and Technology (SMART) Center, with funding from Singapore's National Research Foundation.

arbitrarily small positive constant ϵ), where n is the number of pickup/delivery pairs. From a technical standpoint, the results in this paper leverage a novel connection between the Euclidean Bipartite Matching Problem and the theory of random permutations.

Structure of the paper. This paper is structured as follows. In section II we provide some background on the Euclidean Bipartite Matching Problem and on some notions in probability and group theory. We formally state the stochastic Stacker Crane Problem and the objectives of the paper in section III. In section IV we introduce and analyze the SPLICE algorithm, a polynomial-time, asymptotically optimal algorithm for the SCP, while in section V we present simulation results corroborating our findings. Finally, in section VI, we draw some conclusions and discuss directions for future work.

II. Background material

In this section we summarize the background material used in the paper. Specifically, we review (i) permutations and several of their properties, and (ii) the stochastic Euclidean bipartite matching problem (EBMP).

A. Permutations

A permutation is a rearrangement of the elements of an ordered set $\mathcal{S}$ according to a *bijective* correspondence $\sigma : \mathcal{S} \to \mathcal{S}$. The number of permutations on a set of n elements is given by $n!$, and the set of all permutations over such a set will be denoted by Π_n. As an example, a particular permutation of the set $\{1, 2, 3, 4\}$ is $\sigma(1) = 3$, $\sigma(2) = 1$, $\sigma(3) = 2$, and $\sigma(4) = 4$, which leads to the reordered set $\{3, 1, 2, 4\}$. A permutation can be conveniently represented in a two-line notation, where one lists the elements of $\mathcal{S}$ in the first row and their images in the second row, with the property that a first-row element and its image are in the same column. For the previous example, one would write:

$$\begin{bmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 2 & 4 \end{bmatrix}. \tag{1}$$

The *identity permutation* maps every element of a set $\mathcal{S}$ to itself and will be denoted by σ_1. We will often use the following elementary properties of permutations, which follow from the fact that permutations are bijective correspondences:

- Prop. 1: Given two permutations σ, $\hat{\sigma} \in \Pi_n$, the composition $\sigma(\hat{\sigma})$ is again a permutation.
- Prop. 2: Each permutation $\sigma \in \Pi_n$ has an inverse permutation σ^{-1}, with the property that $\sigma(x) = y$ if and only if $\sigma^{-1}(y) = x$. (Thus, note that $\sigma^{-1}(\sigma) = \sigma_1$.)
- Prop. 3: For any $\hat{\sigma} \in \Pi_n$, it holds $\Pi_n = \{\sigma(\hat{\sigma}),\, \sigma \in \Pi_n\}$; in other words, for a given permutation $\hat{\sigma}$ playing the role of basis, Π_n can be expressed in terms of composed permutations.

A permutation $\sigma \in \Pi_n$ is said to have a *cycle* $\mathcal{L} \subseteq \mathcal{S}$ if the objects in $\mathcal{L}$ form an orbit under the sequence $l_{t+1} = \sigma(l_t)$, i.e.,

$$\sigma(l_t) = l_{t+1} \quad \text{for } t = 1, \dots, T-1, \quad \text{and } \sigma(l_T) = l_1,$$

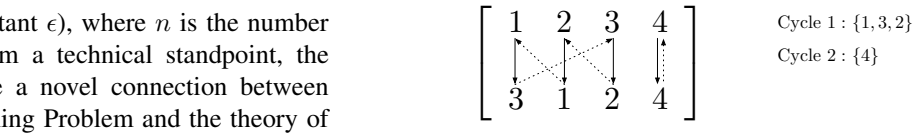


Fig. 1. The two cycles corresponding to the permutation: $\sigma(1) = 3$, $\sigma(2) = 1$, $\sigma(3) = 2$, and $\sigma(4) = 4$. Cycle 1 can equivalently be expressed as $\{2, 1, 3\}$ or $\{3, 2, 1\}$. Apart from this cyclic reordering, the decomposition into disjoint cycles is unique.

where $l_t \in \mathcal{L}$ for all t and T is an integer larger than or equal to one; given a permutation σ, the partition of $\mathcal{S}$ into disjoint cycles is *uniquely* determined apart from a *cyclic* reordering of the elements within each cycle (see figure 1). Henceforth, we denote by $N(\sigma)$ the number of distinct cycles of σ. In the example in equation (1), there are two cycles, namely $\{1, 3, 2\}$, which corresponds to $\sigma(1) = 3$, $\sigma(3) = 2$, $\sigma(2) = 1$, and $\{4\}$, which corresponds to $\sigma(4) = 4$ (see figure 1).

Suppose that all elements of Π_n are assigned probability $1/n!$, i.e.,

$$\mathbb{P}[\sigma] := \mathbb{P}[\text{One selects } \sigma] = \frac{1}{n!}, \qquad \text{for all } \sigma \in \Pi_n.$$

Let N_n denote the number of cycles of a random permutation with the above probability assignment. It is shown in [9] that the number of cycles N_n has expectation $\mathbb{E}[N_n] = \log(n) + O(1)$ and variance $\text{var}(N_n) = \log(n) + O(1)$; here log denotes the natural log.

B. The Euclidean Bipartite Matching Problem

Let $\mathcal{X} = \{x_1, \dots, x_n\}$ and $\mathcal{Y} = \{y_1, \dots, y_n\}$ be two sets of n objects each. The general *bipartite matching problem* (BMP) is to find a permutation $\sigma^* \in \Pi_n$ over the indices $\{1, \dots, n\}$ (not necessarily unique) according to which each y_i is *matched* to $x_{\sigma^*(i)}$ and some cost criterion is minimized. We will often refer to a minimizing permutation σ^* as the *optimal bipartite matching*. We will express the n *matched pairs* as $(x_{\sigma^*(i)},\, y_i)$, for $i = 1, \dots, n$.

The *Euclidean* bipartite matching problem (EBMP), between sets $\mathcal{X}$ and $\mathcal{Y}$ of points in Euclidean space, is to find a permutation σ^* (not necessarily unique) such that the sum of the Euclidean distances between the matched pairs is minimized, i.e.:

$$\sum_{i=1}^{n} \|x_{\sigma^*(i)} - y_i\| = \min_{\sigma \in \Pi_n} \sum_{i=1}^{n} \|x_{\sigma(i)} - y_i\|,$$

where $\|\cdot\|$ denotes the Euclidean norm (see figure 2). Let $Q := (\mathcal{X}, \mathcal{Y})$; we refer to the left-hand side in the above equation as the optimal bipartite matching cost $L_{\text{M}}(Q)$.

The Euclidean bipartite matching problem is generally solved by the "Hungarian" method [10], which runs in $O(n^3)$ time. The $O(n^3)$ barrier was indeed broken by Agarwal et. al [11], who presented a class of algorithms running in $O(n^{2+\epsilon})$, where ϵ is an arbitrarily small positive constant. Additionally, there are also several approximation algorithms: among others, the algorithm presented in [12]

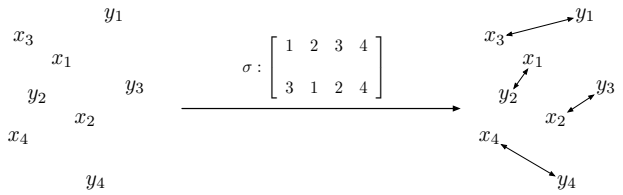


Fig. 2. Example of Euclidean bipartite matching problem whose solution is the permutation: $\sigma(1) = 3$, $\sigma(2) = 1$, $\sigma(3) = 2$, and $\sigma(4) = 4$.

produces a $O(\log(1/\epsilon))$ optimal solution in expected runtime $O(n^{1+\epsilon})$, where, again, ϵ is an arbitrarily small positive constant.

III. Problem Statement

A. The Stochastic (Euclidean) Stacker Crane Problem

Let d be an integer larger than or equal to 2. Let $\mathcal{X}$ be a set of points $X_1, \ldots, X_n$ that are i.i.d. in a compact set $\Omega \subset \mathbb{R}^d$ and distributed according to a density $\varphi_{\mathrm{P}} : \Omega \to \mathbb{R}_{\geq 0}$; let $\mathcal{Y}$ be a set of points $Y_1, \ldots, Y_n$ that are i.i.d. in a compact set $\Omega \subset \mathbb{R}^d$ and distributed according to a density $\varphi_{\mathrm{D}} : \Omega \to \mathbb{R}_{\geq 0}$. We might interpret each pair (X_i, Y_i) as the pick and delivery site, respectively, of the ith customer. Throughout the paper we assume that distributions φ_{P} and φ_{D} are absolutely continuous.

The stochastic Stacker Crane Problem (SCP) is to find a minimum-length tour through the points in $\mathcal{X}$ and $\mathcal{Y}$ with the property that each point X_i (the ith pickup) is immediately followed by the point Y_i (the ith delivery); in other words, the pair (X_i, Y_i) must be visited in consecutive order (see figure 3). We will refer to such a tour as an optimal SCP tour, and to a tour that is not minimum-length but still satisfies the pickup-to-delivery constraints as a SCP tour. In the previous definition, the length of a tour is the sum of all Euclidean distances on the tour. Note that the Stacker Crane Problem is a *constrained* version of the well-known Traveling Salesman Problem.

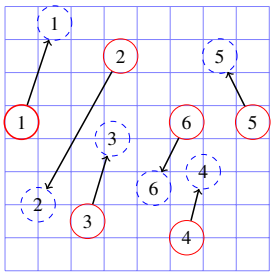
In this paper we aim at solving the following problem:

> **Problem:** Find a polynomial-time algorithm $\mathcal{A}$ for the SCP which is asymptotically optimal almost surely, i.e.
>
> $$\lim_{n \to +\infty} L_{\mathcal{A}} / L^*_{\mathrm{SC}} = 1,$$
>
> where $L_{\mathcal{A}}$ is the length of the SCP tour produced by algorithm $\mathcal{A}$, and L^*_{SC} is the length of the optimal SCP tour.

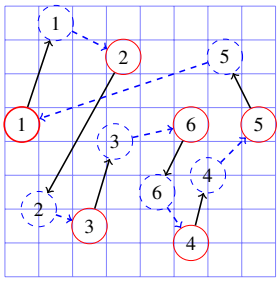
Henceforth, we will refer to the stochastic SCP simply as SCP, with the understanding that all pickup/delivery pairs are generated randomly according to the aforementioned probability model.

IV. An asymptotically optimal polynomial-time algorithm for the SCP

In this section we present an asymptotically optimal, polynomial-time algorithm for the stacker crane problem, which we call SPLICE. The key idea behind the algorithm is



(a) Six pickup/delivery pairs are generated in the Euclidean plane. Solid and dashed circles denote pickup and delivery points, respectively; solid arrows denote the routes from pickup points to their delivery points.



(b) Dashed arrows *combined* with the solid arrows represent a SCP tour.

Fig. 3. Example of Stacker Crane Problem in two dimensions.

to connect the tour from delivery sites back to pickup sites in accordance with an optimal bipartite matching between the sets of pickup and delivery sites. Unfortunately, this procedure is likely to generate a certain number of *disconnected subtours* (see figure 4(b)), and so, in general, the result is not a SCP tour. The key property we will prove is that, perhaps surprisingly, the number of disconnected subtours is "asymptotically small". Then, by using a greedy algorithm to connect such subtours, one obtains an asymptotically optimal solution to the SCP with a polynomial number of operations (since an EBM can be computed in polynomial time).

We start with a formal description of the SPLICE algorithm and a characterization of its computational complexity, then we show the aforementioned property about the number of disconnected subtours, and finally we prove its asymptotic optimality.

A. Algorithm

The algorithm SPLICE is described in pseudo-code on the following page. In line 1, the algorithm $\mathcal{M}$ is *any* algorithm that computes optimal bipartite matchings. After the pickup-to-delivery links and the optimal bipartite matching links are added (lines 2-3), there might be a number of disconnected subtours (they do, however, satisfy the pickup-to-delivery contraints). In such case (i.e., when $N > 1$), links between subtours are added by using a nearest-neighbor rule[1]. Figure 4 shows a sample execution of the algorithm; we refer to the delivery-to-pickup links added in lines 12 and 15 (the green links in figure 4) as *connecting links*, since they connect the subtours. The complexity of SPLICE is dominated by the construction of the optimal bipartite matching, which takes time $O(n^{2+\epsilon})$.

B. Asymptotic number of subtours

The SPLICE algorithm produces, in general (i.e., when $N > 1$), a number of *connecting links* between disconnected

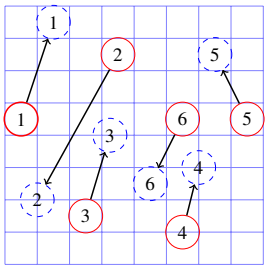
[1]In this paper we use a simple heuristic (from SPLICE line 5) for adding connecting links to form an SCP tour. However, the results of this paper do not depend on this choice, and *any* connecting heuristic can be used.

Algorithm SPLICE

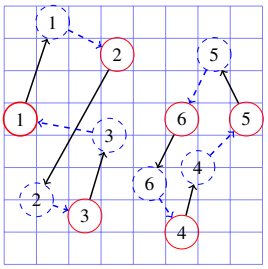
Input: a set of *demands* $\mathcal{S} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$, $n > 1$.

Output: a Stacker Crane tour through $\mathcal{S}$.

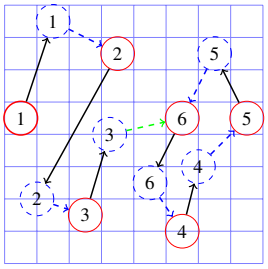
1: **initialize** $\sigma \leftarrow$ solution to Euclidean bipartite matching problem between sets $\mathcal{X} = \{x_1, \ldots, x_n\}$ and $\mathcal{Y} = \{y_1, \ldots, y_n\}$ computed by using a bipartite matching algorithm $\mathcal{M}$.
2: Add the n pickup-to-delivery links $x_i \rightarrow y_i$, $i = 1, \ldots, n$.
3: Add the n matching links $y_i \rightarrow x_{\sigma(i)}$, $i = 1, \ldots, n$.
4: $N \leftarrow$ number of disconnected subtours.
5: **if** $N > 1$ **then**
6: Arbitrarily order the N subtours $\mathcal{S}_j$, $j = 1, \ldots, n$, into an ordered set $\mathcal{S} := \{\mathcal{S}_1, \ldots, \mathcal{S}_N\}$.
7: `base` $\leftarrow$ index of an arbitrary delivery site in $\mathcal{S}_1$.
8: `prev` $\leftarrow$ `base`.
9: **for** $k = 1 \rightarrow N - 1$ **do**
10: Remove link $y_{\texttt{prev}} \rightarrow x_{\sigma(\texttt{prev})}$.
11: `next` $\leftarrow$ index of pickup site in $\mathcal{S}_{k+1}$ that is closest to $y_{\texttt{prev}}$.
12: Add link $y_{\texttt{prev}} \rightarrow x_{\texttt{next}}$.
13: `prev` $\leftarrow \sigma^{-1}$(`next`).
14: **end for**
15: Add link $y_{\texttt{prev}} \rightarrow x_{\sigma(\texttt{base})}$.
16: **end if**



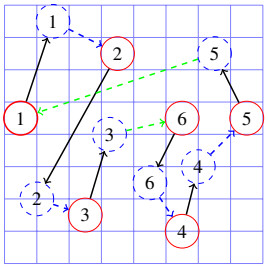
(a) Line 2: 6 pickup-to-delivery links are added.



(b) Line 3: 6 matching links are added. The number of disconnected subtours is $N = 2$.



(c) Line 10. Algorithm state: `prev` = `base` = 3, $k = 1$. The link $y_3 \rightarrow x_1$ is removed, `next` is assigned the value 6, the link $y_3 \rightarrow x_6$ is added, `prev` is assigned the value 5.



(d) Line 15. Algorithm state: `prev` = 5, `base` = 3. The link $y_5 \rightarrow x_1$ is added and the tour is completed.

Fig. 4. Sample execution of the SPLICE algorithm. The solution to the EBMP is $\sigma(1) = 2$, $\sigma(2) = 3$, $\sigma(3) = 1$, $\sigma(4) = 5$, $\sigma(5) = 6$, and $\sigma(6) = 4$. Demands are labeled with integers. Pickup and delivery sites are represented by red and blue circles, respectively. Pickup-to-delivery links are shown as black arrows. Matching links are blue, dashed arrows. Subtour connections are shown as green, dashed arrows. The resulting tour is $1 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 4 \rightarrow 5 \rightarrow 1$.

subtours (see figure 4). The first step to prove asymptotic optimality of the SPLICE algorithm is to characterize the growth order for the number of subtours with respect to n, the size of the problem instance. To this purpose, the following lemma shows the equivalence between the number of subtours N produced by line 3 and the number of cycles for the permutation σ in line 1.

Lemma 4.1 (Permutation cycles and subtours): The number N of subtours produced by the SPLICE algorithm in line 3 is equal to $N(\sigma)$, where $N(\sigma)$ is the number of cycles of the permutation σ computed in line 1.

Proof: Let $\mathcal{D}_k$ be the set of delivery sites for subtour k ($k = 1, \ldots, N$). By construction, the indices in $\mathcal{D}_k$ constitute a cycle of the permutation σ. For example, in figure 4, the indices of the delivery sites in the subtour $x_1 \rightarrow y_1 \rightarrow x_2 \rightarrow y_2 \rightarrow x_3 \rightarrow y_3 \rightarrow x_1$ are $\{1, 2, 3\}$, and they constitute a cycle for σ since $\sigma(1) = 2$, $\sigma(2) = 3$, and $\sigma(3) = 1$. Since the subtours are disconnected, and every index is contained by some subtour, then the sets $\mathcal{D}_k$ ($k = 1, \ldots, N$) represent a partition of $\{1, \ldots, n\}$ into the disjoint cycles of σ. This implies that the number of subtours N is equal to $N(\sigma)$. ■

The above lemma implies that characterizing the number of subtours generated during the execution of the algorithm is *equivalent* to characterizing the number of cycles for the permutation σ. By leveraging the i.i.d. structure in our problem setup, one can intuitively argue that all permutations are equiprobable, and therefore $N_n = N(\sigma_n) = \log(n) + O(1)$ (see section II-A), and $\lim_{n \rightarrow +\infty} N_n / n = 0$. The remainder of this section provides a rigorous proof for this statement.

Let $s = ((x_1, y_1), \ldots, (x_n, y_n))$ (i.e., $s \in \Omega^{2n}$) be a finite *batch* of demands. We may occasionally think of s as a column vector formed from vertical concatenation of $x_1, y_1, \ldots, x_n, y_n$. Thus, we note that Ω^{2n} is a full-dimensional subset of $\mathbb{R}^{d(2n)}$. Let $\Pi^* : \Omega^{2n} \rightarrow 2^{\Pi_n}$ be the *optimal permutation map* that maps a batch $s \in \Omega^{2n}$ into the *set* of permutations that correspond to optimal bipartite matchings (recall that there might be multiple optimal bipartite matchings). Let us denote the set of batches that lead to non-unique optimal bipartite matchings as:

$$\mathcal{Z} := \left\{ s \in \Omega^{2n} \,\middle|\, |\Pi^*(s)| > 1 \right\},$$

where $|\Pi^*(s)|$ is the cardinality of set $\Pi^*(s)$. The behavior of a bipartite algorithm on the set $\mathcal{Z}$ can vary; on the other hand, we now show that set $\mathcal{Z}$ has Lebesgue measure zero, and therefore the behavior of an algorithm on this set is immaterial for our analysis.

Lemma 4.2 (Measure of multiple solutions): The set $\mathcal{Z}$ has Lebesgue measure equal to zero.

Proof: The strategy of the proof is to show that $\mathcal{Z}$ is the subset of a set that has zero Lebesgue measure.

For $\sigma', \sigma'' \in \Pi_n$, $\sigma' \neq \sigma''$, let us define the sets:

$$\mathcal{H}_{\sigma', \sigma''} := \left\{ s \in \Omega^{2n} \,\middle|\, \sum_{i=1}^{n} \|x_{\sigma'(i)} - y_i\| = \sum_{i=1}^{n} \|x_{\sigma''(i)} - y_i\| \right\};$$

let us also define the union of such sets:

$$\mathcal{H} := \bigcup_{\substack{\sigma', \sigma'' \in \Pi_n \\ \sigma' \neq \sigma''}} \mathcal{H}_{\sigma', \sigma''}.$$

The equality constraint in the definition of $\mathcal{H}_{\sigma', \sigma''}$ implies that $\mathcal{H}_{\sigma', \sigma''} \subseteq \mathbb{R}^{d(2n)-1}$, which has zero Lebesgue measure in $\mathbb{R}^{d(2n)}$. Hence, the Lebesgue measure of $\mathcal{H}_{\sigma', \sigma''}$ is zero. Since $\mathcal{H}$ is the union of finitely many sets of measure zero, it has zero Lebesgue measure as well.

We conclude the proof by showing that $\mathcal{Z} \subseteq \mathcal{H}$. Indeed, if $s \in \mathcal{Z}$, there must exist two permutations $\sigma' \neq \sigma''$ such that $\sum_{i=1}^n \|x_{\sigma'(i)} - y_i\| = \min_\sigma \sum_{i=1}^n \|x_{\sigma(i)} - y_i\|$ and $\sum_{i=1}^n \|x_{\sigma''(i)} - y_i\| = \min_\sigma \sum_{i=1}^n \|x_{\sigma(i)} - y_i\|$, i.e., there must exist two permutations $\sigma' \neq \sigma''$ such that

$$\sum_{i=1}^n \|x_{\sigma'(i)} - y_i\| = \sum_{i=1}^n \|x_{\sigma''(i)} - y_i\|,$$

which implies that $s \in \mathcal{H}$. Hence, $\mathcal{Z} \subseteq H$ and, therefore, it has zero Lebesgue measure.

■

In the description of the SPLICE algorithm, $\mathcal{M}$ is an algorithm that computes an optimal bipartite matching (i.e., a permutation that solves an EBMP). According to our definitions, the behavior of such an algorithm can be described as follows: given a batch $s \in \Omega^{2n}$ it computes

$$\mathcal{M}(s) = \begin{cases} \text{unique } \sigma \in \Pi^*(s) & \text{if } s \in \Omega^{2n} \setminus \mathcal{Z}, \\ \text{some } \sigma \in \Pi^*(s) & \text{otherwise.} \end{cases}$$

We want to compute the probability that $\mathcal{M}$ produces as a result the permutation σ; we call such probability $\mathbb{P}[\sigma]$.

Lemma 4.3 (Equiprobability of permutations): The probability that the optimal bipartite matching algorithm $\mathcal{M}$ produces as a result the permutation σ is:

$$\mathbb{P}[\sigma] = \frac{1}{n!}.$$

Proof: We start by observing that it is enough to consider a restricted sample space, namely $\Omega^{2n} \setminus \mathcal{Z}$. Indeed, by the total probability law, and since $\mathcal{Z}$ is a set of zero Lebesgue measure,

$$\mathbb{P}[\sigma] = \mathbb{P}[\mathcal{M}(s) = \sigma |\, s \in \Omega^{2n} \setminus \mathcal{Z}]. \tag{2}$$

For each permutation $\sigma \in \Pi_n$, let us define the set

$$\mathcal{S}_\sigma := \left\{ s \ \in \Omega^{2n} \setminus \mathcal{Z} \,|\, \mathcal{M}(s) = \sigma \right\}.$$

Collectively, sets $\mathcal{S}_\sigma$ form a partition of $\Omega^{2n} \setminus \mathcal{Z}$. This fact, coupled with equation (2), implies

$$\mathbb{P}[\sigma] = \mathbb{P}[s \in \mathcal{S}_\sigma].$$

For a permutation $\sigma \in \Pi_n$, let us define the *reordering function* $g_\sigma : \Omega^{2n} \to \Omega^{2n}$ as the function that maps a batch $s = ((x_1, y_1), \ldots, (x_n, y_n))$ into a batch $s = ((x_{\sigma(1)}, y_1), \ldots, (x_{\sigma(n)}, y_n))$. Let $E_j \in \mathbb{R}^{d \times 2nd}$ be a block row matrix of $2n$ $d \times d$ square blocks whose elements are equal to zero except the $(2j-1)$th block that is identity. Let $F_j \in \mathbb{R}^{d \times 2nd}$ be such a block matrix, but whose elements are all zero except the $2j$th block that is identity. Then in matrix form the reordering function can be written as $g_\sigma(s) = P_\sigma\, s$, where P_σ is the $2nd \times 2nd$ matrix

$$P_\sigma = \begin{bmatrix} E_{\sigma(1)} \\ F_1 \\ E_{\sigma(2)} \\ F_2 \\ \vdots \\ E_{\sigma(n)} \\ F_n \end{bmatrix}.$$

Note that $|\det(P_\sigma)| = 1$ for all permutations σ; also, Prop. 2 of section II-A implies $P_{\sigma^{-1}} = P_\sigma^{-1}$. We now show that $g_{\hat\sigma}(\mathcal{S}_{\hat\sigma}) = \mathcal{S}_{\sigma_1}$ for all permutations $\hat\sigma \in \Pi_n$, recalling that σ_1 denotes the *identity* permutation (i.e., $\sigma(i) = i$ for $i = 1, \ldots, n$):

- $g_{\hat\sigma}(\mathcal{S}_{\hat\sigma}) \subseteq \mathcal{S}_{\sigma_1}$. Let $\hat s \in \mathcal{S}_{\hat\sigma}$. Then by definition $\sum_{i=1}^n \left\|\hat x_{\hat\sigma(i)} - \hat y_i\right\| = \min_{\sigma \in \Pi_n} \sum_{i=1}^n \left\|\hat x_{\sigma(i)} - \hat y_i\right\|$; moreover $\hat\sigma$ is the unique minimizer. We want to show that $g_{\hat\sigma}(\hat s) \in \mathcal{S}_{\sigma_1}$, where $g_{\hat\sigma}(\hat s)$ has the form $((\hat x_{\hat\sigma(1)}, \hat y_1), \ldots, (\hat x_{\hat\sigma(n)}, \hat y_n))$. Let $s = g_{\hat\sigma}(\hat s)$; indeed, σ_1 is an optimal matching of s (by inspection), i.e., $\sigma_1 \in \Pi^*(s)$. Suppose, however, there is another optimal matching $\hat\sigma \neq \sigma_1$ such that $\hat\sigma \in \Pi^*(s)$. Then $\hat{\hat\sigma}(\hat\sigma)$ is an optimal matching of $\hat s$ (Prop. 1); yet this is a contradiction, because $\hat{\hat\sigma}(\hat\sigma) \neq \hat\sigma$. Therefore, we have that $s \in \mathcal{S}_{\sigma_1}$ for all $\hat s \in \mathcal{S}_{\hat\sigma}$.
- $\mathcal{S}_{\sigma_1} \subseteq g_{\hat\sigma}(\mathcal{S}_{\hat\sigma})$. Let $s \in \mathcal{S}_{\sigma_1}$. Then by definition $\sum_{i=1}^n \|x_i - y_i\| = \min_{\sigma \in \Pi_n} \sum_{i=1}^n \left\|x_{\sigma(i)} - y_i\right\|$; moreover σ_1 is the unique minimizer. Note that $g_{\hat\sigma}$ is an injective function (since the determinant of $P_{\hat\sigma}$ is nonzero); let $\hat s$ be the unique batch such that $s = g_{\hat\sigma}(\hat s)$, i.e., $\hat s = ((x_{\hat\sigma^{-1}(1)}, y_1), \ldots, (x_{\hat\sigma^{-1}(n)}, y_n))$ (Prop. 2). We want to show that $\hat s \in \mathcal{S}_{\hat\sigma}$. Because $\sum_{i=1}^n \|x_i - y_i\| = \sum_{i=1}^n \left\|x_{\hat\sigma(\hat\sigma^{-1}(i))} - y_i\right\|$, $\hat\sigma$ is an optimal matching of $\hat s$, i.e., $\hat\sigma \in \Pi^*(\hat s)$. Suppose there is another optimal matching $\hat{\hat\sigma} \neq \hat\sigma$ such that $\hat{\hat\sigma} \in \Pi^*(\hat s)$. Again, this is a contradiction, since $\hat{\hat\sigma}(\hat\sigma^{-1}) \neq \sigma_1$, and σ_1 is the *unique* optimal matching for batch s. We conclude that $\hat s \in \mathcal{S}_{\hat\sigma}$ for all $s \in \mathcal{S}_{\sigma_1}$.

We are now ready to evaluate the probabilities of permutations as follows: First, for any permutation $\hat\sigma$ we have $\mathbb{P}[\hat\sigma] = \mathbb{P}[\hat s \in \mathcal{S}_{\hat\sigma}] = \int_{\hat s \in \mathcal{S}_{\hat\sigma}} \varphi(\hat s) d\hat s$, where $\varphi(\hat s)$ denotes $\prod_{i=1}^n \varphi_{\mathrm{P}}(\hat x_i) \varphi_{\mathrm{D}}(\hat y_i)$. We use variable substitution $s = g_{\hat\sigma}(\hat s) = P_{\hat\sigma} \hat s$ and the property $g_{\hat\sigma}(\mathcal{S}_{\hat\sigma}) = \mathcal{S}_{\sigma_1}$, and we apply the rule of integration by substitution: $\int_{\hat s \in \mathcal{S}_{\hat\sigma}} \varphi(\hat s) d\hat s = \int_{s \in \mathcal{S}_{\sigma_1}} \varphi(P_{\hat\sigma}^{-1} s) \underbrace{|\det(P_{\hat\sigma})|^{-1}}_{=1}\, ds$. Observing that

$$\varphi(P_{\hat\sigma}^{-1} s) = \varphi(P_{\hat\sigma^{-1}} s) = \prod_{i=1}^n \varphi_{\mathrm{P}}(x_{\hat\sigma^{-1}(i)}) \varphi_{\mathrm{D}}(y_i),$$

and that

$$\prod_{i=1}^n \varphi_{\mathrm{P}}(x_{\hat\sigma^{-1}(i)}) \varphi_{\mathrm{D}}(y_i) = \prod_{i=1}^n \varphi_{\mathrm{P}}(x_i) \varphi_{\mathrm{D}}(y_i) = \varphi(s),$$

we obtain

$$\int_{s\in\mathcal{S}_{\sigma_1}} \varphi(P_{\hat{\sigma}}^{-1}s)\ ds = \int_{s\in\mathcal{S}_{\sigma_1}} \varphi(s)\ ds = \mathbb{P}[s\in\mathcal{S}_{\sigma_1}] = \mathbb{P}[\sigma_1].$$

Combining these results, we conclude $\mathbb{P}[\sigma] = \mathbb{P}[\sigma_1]$ for all $\sigma \in \Pi_n$, obtaining the lemma. ∎

Lemmas 4.1 and 4.3 allow us to apply the results in Section II-A and characterize the growth order for the number of subtours.

Lemma 4.4 (Asymptotic number of subtours): Let $d \geq 2$. Let $\mathcal{X}_n$ be a set of points $\{X_1, \ldots, X_n\}$ that are i.i.d. in a compact set $\Omega \subset \mathbb{R}^d$ and distributed according to a density φ_{P}; let $\mathcal{Y}_n$ be a set of points $\{Y_1, \ldots, Y_n\}$ that are i.i.d. in a compact set $\Omega \subset \mathbb{R}^d$ and distributed according to a density φ_{D}. Let N_n be the number of subtours generated by the SPLICE algorithm for a problem instance of size n, i.e., with inputs $\mathcal{X}_n$ and $\mathcal{Y}_n$. Then

$$\lim_{n\to+\infty} N_n/n = 0,$$

almost surely.

Proof: For any $\epsilon > 0$, consider the sequence of events

$$E_n \doteq \Big\{(\mathcal{X}_n, \mathcal{Y}_n) :\ N_n/n > \epsilon\Big\}$$

or, equivalently, $E_n = \Big\{(\mathcal{X}_n, \mathcal{Y}_n) : (N_n - \mathbb{E}N_n) + (\mathbb{E}N_n - \log(n)) + \log(n) > \epsilon n\Big\}$. Now, from lemma 4.1, the number of disconnected subtours is *equal* to the number of cycles in the permutation σ computed by the matching algorithm $\mathcal{M}$ in line 1. Since, by lemma 4.3, all permutations are equiprobable, the number of cycles has expectation and variance both equal to $\log(n)+O(1)$. Therefore, we conclude that N_n has expectation and variance both $\log(n) + O(1)$. Hence, we can rewrite the events E_n as:

$$E_n = \Big\{(\mathcal{X}_n, \mathcal{Y}_n) :\ N_n - \mathbb{E}N_n > \epsilon\, n + o(n)\Big\}.$$

Applying Chebyshev's inequality, we obtain (for n' sufficiently large, yet finite)

$$\sum_{n=0}^{\infty} \mathbb{P}\left[E_n\right] \leq n' + \sum_{n=n'}^{\infty} \frac{\log(n) + O(1)}{\left[\epsilon\, n + o(n)\right]^2}.$$

Since this summation is finite, we can apply the Borel-Cantelli lemma to the sequence of events E_n and conclude that $\mathbb{P}[\limsup_{n\to+\infty} E_n] = 0$. Finally, since ϵ can be chosen arbitrarily small, the upper limit of the claim follows (the lower limit holds trivially). ∎

C. Asymptotic optimality

Having characterized the number of subtours generated by SPLICE, we are now ready to show the asymptotic optimality of the algorithm.

Theorem 4.5: Let $d \geq 2$. Let $\mathcal{X}_n$ be a set of points $\{X_1, \ldots, X_n\}$ that are i.i.d. in a compact set $\Omega \subset \mathbb{R}^d$ and distributed according to a density φ_{P}; let $\mathcal{Y}_n$ be a set of points $\{Y_1, \ldots, Y_n\}$ that are i.i.d. in a compact set $\Omega \subset \mathbb{R}^d$ and distributed according to a density φ_{D}. Then

$$\lim_{n\to+\infty} \frac{L_{\mathrm{SPLICE}}}{L^*_{\mathrm{SC}}} = 1, \quad \text{almost surely.}$$

Proof: Let $Q_n = (\mathcal{X}_n, \mathcal{Y}_n)$. The optimal stacker crane tour through the pickup points $\mathcal{X}_n$ and the delivery points $\mathcal{Y}_n$ is bounded below by

$$L^*_{\mathrm{SC}} \geq \sum_i \|Y_i - X_i\| + L_{\mathrm{M}}(Q_n). \tag{3}$$

On the other hand, the number of connecting links added by the SPLICE algorithm is bounded above by the number of subtours N_n of the optimal bipartite matching, and the length of any connecting link is bounded above by $\max_{x,y\in\Omega} \|x - y\|$. Hence, L_{SPLICE} can be bounded above by

$$\begin{aligned} L_{\mathrm{SPLICE}} &\leq \sum_i \|Y_i - X_i\| + L_{\mathrm{M}}(Q_n) + \max_{x,y\in\Omega} \|x - y\|\, N_n \\ &\leq L^*_{\mathrm{SC}} + \max_{x,y\in\Omega} \|x - y\|\, N_n. \end{aligned}$$

By the strong law of large numbers, $\lim_{n\to+\infty} \sum_i \|Y_i - X_i\|/n = \mathbb{E}\left[\|Y_i - X_i\|\right]$ almost surely. Hence, L^*_{SC} has linear growth (at least). Since $\lim_{n\to+\infty} N_n/n = 0$ (by lemma 4.4), one obtains the claim. ∎

V. Simulation Results

In this section we present simulation results to support the theoretical findings of the paper. Specifically, we discuss (i) rate of convergence to the optimal solution, (ii) the runtime for the SPLICE algorithm, and (iii) the ratio between the runtime of SPLICE and that of an exact algorithm. In all simulations we assume that the pickup/delivery pairs are generated i.i.d. in a unit cube and that φ_{P} and φ_{D} are both uniform. The bipartite matching problem in line 1 of SPLICE is solved using the GNU Linear Programming Toolkit software on a linear program written in MathProg/AMPL. Simulations were run on a laptop computer with a 2.66 GHz dual core processor and 2 GB of RAM.

Figure 5 shows the *factor of optimality* ratios $L_{\mathrm{SPLICE}}/L^*_{\mathrm{SC}}$ with respect to the size of the problem instance n (the number of origin/destination pairs). One can see that the factor of optimality is consistently below 20% even for small problem instances ($n \simeq 10$) and is reduced to $\simeq 5\%$ for $n > 80$. Hence, convergence to the optimal solutions with respect to the problem size is fairly rapid. In practice, one could combine SPLICE with an exact algorithm, and let the exact algorithm compute the solution if n is less than, say, 50, and let SPLICE compute the solution when $n \geq 50$.

Figure 6 shows the runtime T_{SPLICE} of the SPLICE algorithm with respect to the size of the problem instance n (the problem instances are the same as those in figure 5). One can note that even for moderately large problem instances (say, $n \simeq 100$) the runtime is below a second.

Finally, figure 7 shows the *runtime factor* ratios T_{SPLICE}/T^* with respect to the size of the problem instance

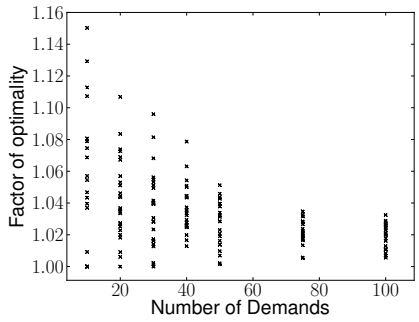


Fig. 5. Factor of optimality for SPLICE as a function of the problem size n. Each column records 25 random observations.

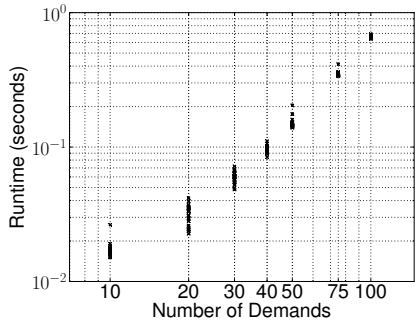


Fig. 6. Runtime of the SPLICE algorithm as a function of the problem size n. Each column records 25 random observations.

n (the problem instances are the same as those in figure 5), where T^* is the runtime of an exact algorithm implemented as an integer linear program. One can observe that the computation of an optimal solution becomes impractical for a number $n \simeq 100$ of origin/destination pairs.

VI. Conclusion

In this paper we presented the SPLICE algorithm, a polynomial-time, asymptotically optimal algorithm for the (stochastic) SCP. This algorithm has complexity of the order $O(n^{2+\epsilon})$ (for arbitrarily small positive constant ϵ) and, in practice, computes solutions within 5% of the optimal for n larger than, say, 100.

This paper leaves numerous important extensions open for further research. First, we are currently developing (asymptotic and almost sure) bounds for the cost of the optimal stochastic SCP solution, and also for the solution delivered by SPLICE. These bounds are instrumental to derive necessary and sufficient conditions for the existence of stable routing policies for a dynamic version of the stochastic SCP, whereby the pickup and delivery sites are dynamically generated by an exogeneous process. Second, we are interested in precisely characterizing the convergence rate to the optimal solution, and in addressing the more general case where the pickup and delivery locations are statistically correlated, Third, we plan to extend the SPLICE algorithm and its analysis to the case of multiple vehicles.

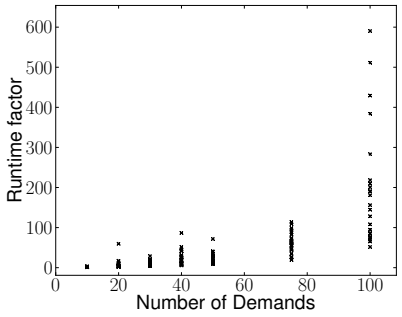


Fig. 7. Runtime factors for the SPLICE algorithm as a function of the problem size n. Each column records 25 random observations.

Fourth, while in the SCP the servicing vehicle is assumed to be omnidirectional (i.e., sharp turns are allowed), we hope to develop approximation algorithms for the SCP where the vehicle has differential motion constraints (e.g., bounded curvature), as is typical, for example, with unmanned aerial vehicles. In addition to these natural extensions, we hope that the techniques introduced in this paper (i.e., coupling the EBMP with the theory of random permutations) may come to bear in the stochastic setting for other hard combinatorial problems.

References

[1] G. Berbeglia, J.-F. Cordeau, I. Gribkovskaia, and G. Laporte. Static pickup and delivery problems: a classification scheme and survey. *TOP*, 15:1–31, 2007. 10.1007/s11750-007-0009-0.

[2] J. F. Cordeau, G. Laporte, J. Y. Potvin, and M.W.P. Savelsbergh. Transportation on demand. In C. Barnhart and G. Laporte, editors, *Transportation, Handbooks in Operations Research and Management Science*, volume 14, pages 429–466. Elsevier, Amsterdam, The Netherlands, 2007.

[3] W. J. Mitchell, C. E. Borroni-Bird, and L. D. Burns. *Reinventing the Automobile*. MIT Press, 2010.

[4] Greg N. Frederickson and D. J. Guan. Preemptive ensemble motion planning on a tree. *SIAM Journal on Computing*, 21(6):1130–1152, 1992.

[5] G. N. Frederickson and D. J. Guan. Nonpreemptive ensemble motion planning on a tree. *J. Algorithms*, 15:29–60, July 1993.

[6] A. Coja-Oghlan, S. O. Krumke, and T. Nierhoff. A heuristic for the stacker crane problem on trees which is almost surely exact. *Journal of Algorithms*, 61(1):1 – 19, 2006.

[7] G. N. Frederickson, M. S. Hecht, and C. E. Kim. Approximation algorithms for some routing problems. *Foundations of Computer Science, Annual IEEE Symposium on*, 0:216–227, 1976.

[8] A. Coja-Oghlan, S. O. Krumke, and T. Nierhoffi. Scheduling a server on a caterpillar network - a probabilistic analysis. In *Proceedings of the 6th Workshop on Models and Algorithms for Planning and Scheduling Problems*, 2003.

[9] L. A. Shepp and S. P Lloyd. Ordered cycle lengths in a random permutation. *Transactions of the American Mathematical Society*, 121(2):340–357, 1966.

[10] H. W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2(1-2):83–97, 1955.

[11] Pankaj K. Agarwal, Alon Efrat, and Micha Sharir. Vertical decomposition of shallow levels in 3-dimensional arrangements and its applications. In *Proceedings of the eleventh annual symposium on Computational geometry*, pages 39–50, Vancouver, British Columbia, Canada, 1995. ACM.

[12] P. Agarwal and K. Varadarajan. A near-linear constant-factor approximation for euclidean bipartite matching? In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 247–252, Brooklyn, New York, USA, 2004. ACM.