2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Minimum Time Point Assignment for Coverage by Two Constrained Robots

Nilanjan Chakraborty
Rensselaer Polytechnic Inst.
Troy, New York 12180
chakrn2@cs.rpi.edu

Srinivas Akella
Rensselaer Polytechnic Inst.
Troy, New York 12180
sakella@cs.rpi.edu

John T. Wen
Rensselaer Polytechnic Inst.
Troy, New York 12180
wenj@rpi.edu

***Abstract*— This paper focuses on the assignment of discrete points to two robots, in the presence of geometric and kinematic constraints between the robots. The individual points have differing processing times, and the goal is to identify an assignment of points to the robots so that the total processing time is minimized. The assignment of points to the robots is the first step in the path generation process for the robots. This work is motivated by an industrial microelectronics manufacturing system with two robots, with square footprints, that are constrained to translate along a common line while satisfying proximity and collision avoidance constraints. The N points lie on a planar base plate that can translate along the plane normal to the direction of motion of the robots. The geometric constraints on the motions of the two robots lead to constraints on points that can be processed simultaneously. We show that the point assignment for processing problem can be converted to a maximum weighted matching problem on a graph and solved optimally in $O(N^3)$ time. Since this is too slow for large datasets, we present a $O(N^2)$ time greedy algorithm and prove that the greedy solution is within a factor of $3/2$ of the optimal solution. Finally, we provide computational results for the greedy algorithm on typical industrial datasets.**

I. Introduction

Robotic point set coverage tasks occur in a variety of application domains like electronic manufacturing (laser drilling [2], inspection [1], circuit board testing [13], [4]), automobile spot welding [9], and data collection in sensor networks [11]. The multi-robot point set coverage task that we consider in this paper has the following characteristics: (a) the robots have to spend some time at each point to complete a task (we call this time *processing time* of each point) and (b) they have to satisfy given geometric constraints while covering the point set. The objective is to assign the points to the robots and find an order of processing the points to minimize the overall time required. Our work is motivated by a laser drilling system shown in Figure 1. Here we need to process (drill) a set of points by a system of $K(=2)$ robots. The architecture of the machine imposes the following geometric constraints: (a) at any instant of time, each robot can process exactly one point within a square region in the plane (called *processing footprint*) although there may be several points within the region, (b) the robots are constrained to move along a line while avoiding collisions, and (c) the points lie on a base plate that can translate along the y-axis. The processing time at each point may differ depending on the radius of the hole (e.g., in drilling by trepanning, where the diameter of the hole to be drilled is larger than that of the laser beam).

We focus here on the assignment of points to the two robots such that the processing operations are parallelized as much as possible while respecting the geometric constraints. This subproblem is a step to our ultimate objective of minimizing the overall time required to process all the points, including the travel time. That overall solution needs both generation of paths and trajectories for the robots, and is beyond the scope of this paper. We previously considered the path planning problem (i.e., the problem of splitting the points, assigning them to the robots, and finding the order of processing the points) for multiple robot point set coverage tasks with the assumption that all points have the same processing time [2]. We showed that for two robots the splitting problem is equivalent to a maximum cardinality matching problem on a graph and can thus be solved optimally in $O(N^3)$ time in general [5], [7], where N is the number of points to be processed.

The solution for the splitting problem presented in [2] is not optimal when the processing times are different. In this paper we show that when the processing times of the points are different, the splitting problem can be formulated as a maximum weighted matching problem (MWM) on a suitably constructed weighted graph. Thus, the splitting problem can be solved in $O(N^3)$ time in general [5], [7]. However, this is not suitable for large datasets (in our applications, approximately 10^5 points). Therefore we present a greedy algorithm that takes $O(N^2)$ time. We prove that the ratio of the processing time obtained with our greedy algorithm to that of the optimal processing time is within a factor of $3/2$. We also provide a simple example to show that this is a tight bound. Finally, for the example setup shown in Figure 1, we provide computational results showing that the greedy algorithm gives solutions very close to the optimal solution on typical industrial datasets.

II. Related Literature

The path planning problem for covering a point set in minimum time by multiple constrained robots is NP-hard and can be divided into a splitting (and assignment) problem and ordering problem [2]. In [2] we showed that when the processing times of the points are identical, the splitting problem can be solved optimally by converting it to a

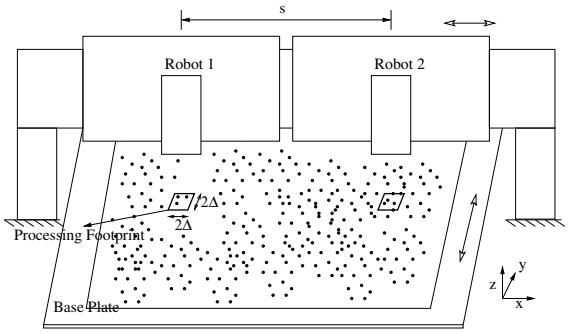

Fig. 1. Schematic sketch of a 2-robot system used to process points in the plane. The robots translate along x-axis and the base plate translates along y-axis. The square of length 2Δ is the processing footprint for each robot.

maximum cardinality matching (MCM) problem on a graph. The ordering problem can be then solved by formulating it as a Traveling Salesman Problem (TSP). When the processing times of the points differ, the MCM solution is not optimal for the splitting problem. In this paper, we provide solutions to the splitting problem when the processing times are different.

The minimum makespan scheduling problem [8] problem is closely related to our splitting problem. There are various versions of the minimum makespan scheduling problem [8]. The problem closest to our interest is minimum makespan scheduling with precedence constraints. This is a NP-hard problem, even for two machines. However, there are two important distinctions between the minimum makespan problem with precedence constraints and our minimum time assignment problem with geometric constraints.

- Minimum makespan scheduling problem has precedence constraints, which means that if two jobs are constrained, then one cannot be started before the other is finished. On the other hand, we have a geometric constraint that only states that two jobs cannot be done simultaneously.
- In the basic version of the minimum makespan scheduling we can assign a new job to a machine when it is finished with the currently assigned job. However here we can start a new job only when both the jobs are finished. This is because our problem occurs as a subproblem of a more general problem where the two robots need to move from one point to another before processing them and also maintain the geometric feasibility conditions, for example, no-collision conditions.

Although the minimum makespan problem with precedence constraints is NP-hard, our problem has a polynomial time solution. We convert the problem to a maximum weighted matching on a suitably constructed graph. Maximum weighted matching is a well studied problem and can be solved in $O(N^3)$ time (Edmonds [3]) where N is the number of vertices of the graph. There are also algorithms that are slightly faster ($e.g.$, Micali and Vazirani's $O(\sqrt{|V|}|E|)$ algorithm [6]).

III. Splitting Problem

In this section we provide solution algorithms for the splitting problem. The splitting problem consists of assigning the set of points to each robot so as to minimize the total processing time while respecting the geometric constraints. A solution to this problem ensures maximum parallelization of the processing operations. To be concrete, we will use the geometric constraints for the problem shown in Figure 1. However, the results are valid for constraints specified by other types of metrics and in higher dimensional spaces. We call a pair of points a *compatible pair (of points)* if they can be processed together while respecting the geometric constraints. Any two compatible pairs are called a *disjoint compatible pair* if the points belonging to the two pairs are distinct. In a compatible pair of points, the point having the larger processing time is called the *dominant* point and the point having a smaller processing time is called the *non-dominant* point. The formal statement for this problem is:

Problem Statement: Let $S = \{\mathbf{p}_i\} = \{(x_i, y_i)\}$, $i = 1, 2, \ldots N$, be a set of points in $\mathbb{R}^2$ with the processing time of point $\mathbf{p}_i$ being t_i. Let P be a set of *ordered* subsets of S of size less than or equal to 2 that partitions S, $i.e.$, $P = \{(\mathbf{p}_i, \mathbf{p}_j)\} \bigcup \{(\mathbf{p}_k, *)\} \bigcup \{(*, \mathbf{p}_l)\}$, $i, j, k, l \in \{1, 2, \ldots, N\}, i \neq j \neq k \neq l$, where $*$ denotes a *virtual point* and any pair $(\mathbf{p}_i, \mathbf{p}_j) \in P$ respects geometric constraints of the form $f(x_i, y_i, x_j, y_j) \leq 0$. For the system shown in Figure 1 the constraints are the following:

$$\begin{aligned} |x_i - x_j| &\geq s_{\min} - 2\Delta \\ |y_i - y_j| &\leq 2\Delta \end{aligned} \tag{1}$$

where $s_{\min}$ is the minimum allowable distance between the two robots and $2\Delta \times 2\Delta$ is the processing footprint of each robot. Find such a P that minimizes the cost given by

$$C = \sum_{\text{all pairs}} \max(t_i, t_j) + \sum_{\text{all singletons}} t_k \tag{2}$$

The ordered pair $(\mathbf{p}_i, \mathbf{p}_j)$ denotes that $\mathbf{p}_i$ is assigned to robot 1 and $\mathbf{p}_j$ to robot 2. Moreover $(\mathbf{p}_k, *)$ denotes that $\mathbf{p}_k$ is a singleton assigned to robot 1, while $(*, \mathbf{p}_l)$ denotes that $\mathbf{p}_l$ is a singleton assigned to robot 2. In Equation 1, the constraint on the $x-$coordinates ensure collision avoidance between the robots. The constraint on the $y-$coordinates indicate that the robots are constrained to move along the $x-$axis but have a square footprint for processing.

Before formulating the problem as a MWM on a graph we define the related terms from matching theory on graphs.

Definition Let $G = (V, E)$ be a weighted graph where V is the set of vertices and E is the set of edges. A set $M \subseteq E$ is called a *matching* if no two edges in M have a vertex in common. M is called a *maximal matching* if there is no matching M' such that $M \subset M'$. M is called a *maximum cardinality matching* (MCM) if it is a maximal matching of maximum cardinality. M is called a *maximum weighted matching* (MWM) if the sum of weights of the edges in M is maximum among all matchings.

A. Optimal Algorithm for Splitting

The cost of any partition of S given by Equation 2 can be rewritten as follows:

$$\begin{aligned} C &= \sum_{\text{all pairs}} \max(t_i, t_j) + \sum_{\text{all singletons}} t_k, \quad (i \neq j \neq k) \\ &= \sum_{\text{all pairs}} (t_i + t_j - \min(t_i, t_j)) + \sum_{\text{all singletons}} t_k \\ &= \sum_{\text{all points}} t_k - \sum_{\text{all pairs}} \min(t_i, t_j), \quad (i \neq j) \end{aligned}$$

Since the sum of the processing times of all points is constant, the problem of minimizing C is equivalent to maximizing $C_1 = \sum_{\text{all pairs}} \min(t_i, t_j)$, i.e., finding a subset of compatible pairs of points such that the sum of the processing times of the non-dominant points (in each pair) is maximized. The solution to this problem is given by the maximum weight matching on a weighted graph. The construction of the graph is as follows: Let $G = (V, E)$ be a weighted graph, with the set of vertices $V = \{1, 2, \ldots, N\}$ corresponding to the points. An edge (i, j) exists between two vertices if the corresponding points $\mathbf{p}_i, \mathbf{p}_j$ are compatible, i.e., they satisfy Equation 1. The weight w_{ij} on each edge is given by $\min(t_i, t_j)$. Thus a MWM on this graph gives a set of disjoint compatible pairs where the sum of the processing times of non-dominant points is maximized.

The MWM problem on a graph is a well known combinatorial optimization problem and can be solved in $O(N^3)$ time (Edmonds [3]). However, there are applications where N can be quite large, and consequently, the matching algorithm is not practical. For example, N can be of the order of 10^5 in the laser drilling application of Figure 1. Hence, we provide a $O(N^2)$ time greedy algorithm that gives a suboptimal solution with the worst case approximation ratio of $3/2$. Note that there are deterministic constant factor approximation algorithms in the weighted matching literature that have running time linear in the number of edges, i.e., $O(N^2)$ in the worst case, (see [12], and references therein). The simplest such algorithm has a worst case approximation ratio of 2 which implies that the ratio of the value of C_1 in the optimal matching to that in the obtained matching is upper bounded by 2. For our problem it means that the ratio of the sum of the processing times of the non-dominant points in the optimal solution to that of the greedy solution is less than or equal to 2. This implies that the cost given by Equation 2 is within a factor of $3/2$ of the optimal cost. However, we can construct simple examples on datasets of 4 points where our proposed algorithm performs better than this graph matching approximation algorithm. There are more sophisticated $O(N^2)$ time algorithms that have an approximation ratio arbitrarily close to $4/3$ for our problem. However, the implementation of such algorithms are more complicated and it is not clear whether those algorithms have been implemented in practice. Therefore, we present a simple greedy algorithm, not requiring explicit construction of the compatibility graph, whose total cost is provably within $3/2$ of the optimal cost.

B. Greedy Algorithm

For our problem, the trivial algorithm where each point is processed individually takes time at most twice that of the optimal solution for the two robots (if there were k robots this bound would be k). Thus any useful algorithm should have a theoretical worst case bound less than 2. When there are no constraints between the points the optimal algorithm to pair up the points is to sort the points in the order of decreasing processing times and pair up consecutive points (starting from the top) in this sorted list. When there are constraints on the points that can be paired, this algorithm is not optimal. However, we prove that it is within a factor of $3/2$ of the optimal solution. The basic algorithm is:

1) Sort the points in order of decreasing processing times.
2) Start with the point having largest processing time. Pair each point with a compatible point (that is not already paired) that has the largest processing time. If there is more than one compatible point with same (largest) processing time choose one at random. If there is no compatible point available make the point a singleton.

The cost of this algorithm is determined by the second step above. A straightforward implementation compares each point to every other point in the worst case and thus the algorithm has a worst case running time of $O(N^2)$.

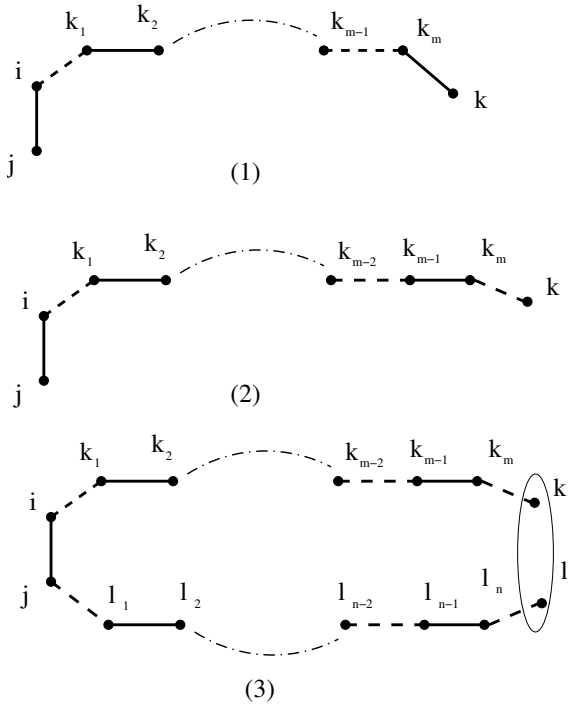

Fig. 2. (1) Schematic sketch of case 1 (2) Schematic sketch of case 2 (3) Schematic sketch of case 3. The bold lines are the pairs in the greedy solution and the dotted lines are the pairs in the optimal solution

Lemma 3.1: Let T_G be the total processing time when the points are paired according to the greedy algorithm and T_O be the optimal processing time. Then $T_G/T_O \leq 3/2$.

Proof: Let the set P_G be the solution of the greedy algorithm and P_O be the solution of the optimal algorithm. The ratio

of the greedy cost to optimal cost can be written as

$$\frac{T_G}{T_O} = \frac{\sum_{P_G} \max(t_i, t_j) + \sum_{P_G} t_k}{\sum_{P_O} \max(t_i, t_j) + \sum_{P_O} t_k} \tag{3}$$

Let $P_G^{'} \subseteq P_G$ and $P_O^{'} \subseteq P_O$ be subsets of the two solutions that contain the same points. We partition the whole solution set into such partial subsets. Let $(\mathbf{p}_i, \mathbf{p}_j) \in P_G^{'}$ be a pair in the greedy solution (w.l.o.g. assume t_i is the greatest element in that set; we will explain later why we do not consider that a singleton can be the greatest element). Then, it suffices to consider the three forms for the sets $P_G^{'}$ and $P_O^{'}$ shown in Figure 2. For brevity, we use the indices of the points to denote the points, for example, $(\mathbf{p}_i, \mathbf{p}_j)$ is denoted by (i, j).

1) We have a sequence in the optimal pairing given by $j, (i, k_1), (k_2, k_3), \ldots, (k_{m-1}, k_m), k$ (m odd) with the corresponding pairing in the greedy solution given by $(i, j), (k_1, k_2), \ldots, (k_m, k)$, i.e., there are two singletons in optimal solution not present in greedy solution.
2) We have a sequence in the optimal pairing given by $j, (i, k_1), (k_2, k_3), \ldots, (k_m, k)$ with the corresponding pairing in the greedy solution given by $(i, j), (k_1, k_2), \ldots, (k_{m-1}, k_m), k$, where m is even.
3) We have a sequence in the optimal pairing given by $(i, k_1), (k_2, k_3), \ldots, (k_m, k), (j, l_1), (l_2, l_3), \ldots, (l_m, l)$ with the corresponding pairing in the greedy solution given by $(i, j), (k_1, k_2), \ldots, (k_{m-1}, k_m), k, (l_1, l_2), \ldots, (l_{m-1}, l_m), l$, i.e., there are two singletons in the greedy solution that are not present in the optimal solution.

Note that we can have at most two singletons in $P_G^{'}$ that may not be a singleton in $P_O^{'}$. So, either we can reach all singletons in the greedy set by this construction or they are present as singletons in the optimal solution also. The first case is of interest to us where, if the singleton had the highest cost it would be paired up (by the nature of our greedy algorithm) because it has a feasible partner. Thus the singleton in $P_G^{'}$ cannot have the highest cost. We now show that for each case above the partial greedy cost (T_G^p) is within a factor of $3/2$ of the partial optimal cost (T_O^p).

Case 1: The ratio of partial greedy cost to optimal cost is

$$\frac{T_G^p}{T_O^p} = \frac{t_i + \sum_{u=1}^{(m-1)/2} \max(t_{k_{2u-1}}, t_{k_{2u}}) + \max(t_{k_m}, t_k)}{t_i + t_j + t_k + \sum_{u=1}^{(m-1)/2} \max(t_{k_{2u}}, t_{k_{2u+1}})}$$

If we assume $t_{k_1} \geq t_{k_2}, t_{k_3} \geq t_{k_4}, \ldots, t_{k_m} \geq t_k$,

$$\frac{T_G^p}{T_O^p} = \frac{t_i + t_{k_1} + \sum_{u=1}^{(m-1)/2} t_{k_{2u+1}}}{t_i + t_j + t_k + \sum_{u=1}^{(m-1)/2} \max(t_{k_{2u}}, t_{k_{2u+1}})} \tag{4}$$

As $t_j \leq t_{k_1}$ (otherwise $P_G^{'}$ would have the pair (i, k_1)) and $\max(t_{k_{2u}}, t_{k_{2u+1}}) \geq t_{k_{2u+1}}$), all the terms in the numerator and denominator of Equation 4 are identical, with t_k as an extra term in the denominator implying that $T_G^p/T_O^p \leq 1$, i.e., the greedy cost is less than the optimal cost. Therefore the processing times of the greedy pair cannot be of the above form. Similarly, we can show that $t_{k_1} \leq t_{k_2}, t_{k_3} \leq t_{k_4}, \ldots, t_{k_m} \leq t_k$ is not possible. Thus, the sequence in $P_G^{'}$ will have alternating subsequences where the first point dominates and then the second point dominates and so forth.

Now, assume that the greedy pairs satisfy $t_{k_1} \leq t_{k_2}, t_{k_3} \leq t_{k_4}, \ldots, t_{k_{l-1}} \leq t_{k_l}$, for some $l < m$ and $t_{k_{l+1}} \geq t_{k_{l+2}}$, i.e., the pairs have their second point dominating upto the pair (k_{l-1}, k_l). We do not assume anything about rest of the points. We develop the proof in two steps. We first show that the ratio of the two partial costs upto this pair is less than $3/2$. We then argue that if there are more such changes in the position of the dominant points in the greedy pairs the ratio of partial costs upto those pairs will also be less than $3/2$. Thus we can extend this scheme to the whole set $P_G^{'}$ and say that the cost $T_G^p/T_O^p \leq 3/2$. We start with the first step. The ratio of the costs upto the point where there is the first change in dominating point is

$$r = \frac{t_i + t_{k_2} + \cdots + t_{k_{l-2}} + t_{k_l} + t_{k_{l+1}}}{t_i + t_j + \cdots + \max(t_{k_{l-2}}, t_{k_{l-1}}) + \max(t_{k_l}, t_{k_{l+1}})}$$

Without loss of generality assume $t_{k_l} \geq t_{k_{l+1}}$ (proof with $t_{k_l} \leq t_{k_{l+1}}$ is analogous). Then $t_{k_{l-1}} \geq t_{k_{l+1}}$, otherwise (k_l, k_{l+1}) would be in $P_G^{'}$ instead of (k_{l-1}, k_l). Using $\max(t_{k_{2u}}, t_{k_{2u+1}}) \geq t_{k_{2u}}$ for all $u < (l-2)/2$ in the denominator and $C = \sum_{u=1}^{(l-3)/2} t_{k_{2u}}$, we have

$$\begin{aligned} r &\leq \frac{t_i + t_{k_{l-2}} + t_{k_l} + t_{k_{l+1}} + C}{t_i + t_j + \max(t_{k_{l-2}}, t_{k_{l-1}}) + t_{k_l} + C} \\ &\leq \frac{t_{k_{l-2}} + t_{k_l} + t_{k_{l+1}}}{\max(t_{k_{l-2}}, t_{k_{l-1}}) + t_{k_l}} \\ &\leq \frac{t_{k_{l-1}} + t_{k_l} + t_{k_{l+1}}}{t_{k_{l-1}} + t_{k_l}} \quad \text{assume w.l.o.g } t_{k_{l-1}} \geq t_{k_{l-2}} \\ &= 1 + \frac{t_{k_{l+1}}}{(t_{k_{l-1}} + t_{k_l})} \leq \frac{3}{2} \quad (\because t_{k_{l+1}} \leq t_{k_{l-1}} \leq t_{k_l}) \end{aligned} \tag{5}$$

If $t_{k_{l-1}} \leq t_{k_{l-2}}$, we could replicate the same arguments substituting $t_{k_{l-2}}$ in place of $t_{k_{l-1}}$. When the sequence in $P_G^{'}$ changes from a second point dominating pair to a first point dominating pair there is only slight increase in the cost of the denominator and consequently the bounds are satisfied. However, when the sequence again changes to a second point dominating pair, we can write the partial sum similar to the above derivation. Let the change be at k_n, i.e., $t_{k_{n+1}} \geq t_{k_{n+2}}$. The second line of equation 5 is then

$$r \leq \frac{t_{k_{l-2}} + t_{k_l} + t_{k_{l+1}} + t_{k_{n-2}} + t_{k_n} + t_{k_{n+1}}}{\max(t_{k_{l-2}}, t_{k_{l-1}}) + t_{k_l} + \max(t_{k_{n-2}}, t_{k_{n-1}}) + t_{k_n}}$$

with $t_{k_{n+1}} \leq t_{k_{n-1}} \leq t_{k_n}$. The next steps are analogous to the steps in equation 5. We can proceed likewise until the whole set $P_G^{'}$ is covered and the approximation ratio is thus less than $3/2$.

Case 2: Assume that $t_{k_1} \leq t_{k_2}, t_{k_3} \leq t_{k_4}, \ldots, t_{k_{m-1}} \leq t_{k_m}$ in $P_G^{'}$ w.l.o.g.; if this were not true we could have proceeded as described in case 1 above. We also have $t_k \leq t_{k_{m-1}} \leq t_{k_m}$, otherwise $P_G^{'}$ would have the pair (k_m, k) instead of k_{m-1}, k_m. The ratio of the cost of $P_G^{'}$ to $P_O^{'}$ is less than $3/2$ as shown below. The second step is obtained by using the assumption above in the numerator and using the fact that $\max(a, b) \geq a, \forall a, b > 0$ in the denominator.

The fourth step is obtained from the third step using the fact that if $a/b > 1$ and $c/d < 1$ then $(a+c)/(b+d) < a/b$.

$$\begin{aligned}
&\frac{t_i + \sum_{u=1}^{m/2} \max(t_{k_{2u-1}}, t_{k_{2u}}) + t_k}{t_i + \sum_{u=1}^{(m-2)/2} \max(t_{k_{2u}}, t_{k_{2u+1}}) + \max(t_k, t_{k_m}) + t_j} \\
\leq &\frac{t_i + \sum_{u=1}^{m/2} t_{k_{2u}} + t_k}{t_i + \sum_{u=1}^{m/2} t_{k_{2u}} + t_j} \\
\leq &\frac{t_i + t_{k_m} + t_k + C}{t_i + t_{k_m} + t_j + C} \quad (C = \sum_{u=1}^{m/2-1} t_{k_{2u}}) \\
\leq &\frac{t_i + t_{k_m} + t_k}{t_i + t_{k_m}} \\
= &1 + \frac{t_k}{(t_i + t_{k_m})} \leq \frac{3}{2} \quad (\because t_k \leq t_{k_m}, t_k \leq t_i)
\end{aligned}$$

Case 3: Here, it may be possible that (k, l) form a pair in the greedy solution (see Figure 2 botom). However, for worst case analysis, it suffices to look at the case when k and l are singletons. Here again we assume that $t_{k_1} \leq t_{k_2}, t_{k_3} \leq t_{k_4}, \ldots t_{k_{m-1}} \leq t_{k_m}$, $t_{l_1} \leq t_{l_2}, t_{l_3} \leq t_{l_4}, \ldots, t_{l_{n-1}} \leq t_{l_n}$ without any loss of generality. We also have $t_k \leq t_{k_{m-1}}$ and $t_l \leq t_{l_{n-1}}$ because (k_{m-1}, k_m) and (l_{n-1}, l_n) are the greedy pairs. The partial costs T_O^p and T_G^p are then

$$\begin{aligned}
T_G^p &= t_i + \sum_{u=1}^{m/2} \max(t_{k_{2u-1}}, t_{k_{2u}}) + \sum_{u=1}^{n/2} \max(t_{l_{2u-1}}, t_{l_{2u}}) \\
&\quad + t_k + t_l = t_i + \sum_{u=1}^{m/2} t_{k_{2u}} + \sum_{u=1}^{n/2} t_{l_{2u}} + t_k + t_l \\
T_O^p &= t_i + \sum_{u=1}^{(m-2)/2} \max(t_{k_{2u}}, t_{k_{2u+1}}) + \max(t_{k_m}, t_k) \\
&\quad + \max(t_j, t_{l_1}) + \sum_{u=1}^{(n-2)/2} \max(t_{l_{2u}}, t_{l_{2u+1}}) + \max(t_{l_n}, t_l) \\
&\quad \geq t_i + \sum_{u=1}^{m/2} t_{k_{2u}} + \sum_{u=1}^{n/2} t_{l_{2u}} + t_j
\end{aligned} \tag{6}$$

The ratio of the partial costs is then

$$\begin{aligned}
\frac{T_G^p}{T_O^p} &\leq \frac{t_i + \sum_{u=1}^{m/2} t_{k_{2u}} + \sum_{u=1}^{n/2} t_{l_{2u}} + t_k + t_l}{t_i + \sum_{u=1}^{m/2} t_{k_{2u}} + \sum_{u=1}^{n/2} t_{l_{2u}} + t_j} \\
&= \frac{t_{k_m} + t_{k_{m-1}} + t_{l_n} + t_{l_{n-1}} + t_k + t_l + C}{t_{k_m} + t_{k_{m-1}} + t_{l_n} + t_{l_{n-1}} + t_j + C} \\
&\quad (\text{where } C = t_i + \sum_{u=1}^{m/2-2} t_{k_{2u}} + \sum_{u=1}^{n/2-2} t_{l_{2u}}) \\
&\leq \frac{t_{k_m} + t_{k_{m-1}} + t_{l_n} + t_{l_{n-1}} + t_k + t_l}{t_{k_m} + t_{k_{m-1}} + t_{l_n} + t_{l_{n-1}}} \\
&= 1 + \frac{t_k + t_l}{t_{k_m} + t_{k_{m-1}} + t_{l_n} + t_{l_{n-1}}} \leq \frac{3}{2} \\
&\quad \text{Using } t_k \leq t_{k_{m-1}} \leq t_{k_m},\ t_l \leq t_{l_{n-1}} \leq t_{l_n}
\end{aligned}$$

The greedy solution set can be represented as an union of disjoint sets of the forms described by the 3 cases above and each partial sum is less than $3/2$ times the corresponding optimal partial cost. Therefore, we have $T_G \leq 3/2\ T_O$. ■

The above bound is a tight bound. We give an example on 4 points where the cost of the greedy algorithm is $3/2$ times the cost of the optimal algorithm. Let the 4 points $\{i, j, k, l\}$ have unit processing times and the set of compatible points be $\{(i,j), (i,k), (i,l), (j,k)\}$. The greedy algorithm may pick the pair (i, j) since all the pairs have equal processing times. Thus, the greedy solution is $\{(i,j), k, l\}$ with cost 3. The optimal cost is 2 with solution $\{(i,l), (j,k)\}$.

In the preceding discussion we gave algorithms to split the points among the two robots. However, we did not make an explicit assignment of the points to the robots. If the geometric constraints are such that one robot is always constrained to be on the left of other (as in Figure 1) we can always assign the point in the pair with smaller $x-$coordinate to one robot and its partner to the other. The ordering problem can then be set up as a multi-dimensional TSP in the pair space [2]. However, if the geometric constraints are different, for example, the constraints are based on maintaining a minimum (or maximum) Euclidean inter-robot distance, then the above scheme may not be appropriate. In this case, we need to solve the assignment and ordering problems simultaneously. Let $(\mathbf{p}_i, \mathbf{p}_j)$ and $(\mathbf{p}_k, \mathbf{p}_l)$ be two pairs and the robots have to move from one pair to the other. The question we need to answer here is: Does the robot at $\mathbf{p}_i$ or the one at $\mathbf{p}_j$ move to $\mathbf{p}_k$? If $\mathbf{p}_i$ moves to $\mathbf{p}_k$ and $\mathbf{p}_j$ moves to $\mathbf{p}_l$ the travel cost is $d_1 = \max\{d(\mathbf{p}_i, \mathbf{p}_k), d(\mathbf{p}_j, \mathbf{p}_l)\}$. On the other hand if $\mathbf{p}_i$ moves to $\mathbf{p}_l$ and $\mathbf{p}_j$ moves to $\mathbf{p}_k$ the travel cost is $d_2 = \max\{d(\mathbf{p}_i, \mathbf{p}_l), d(\mathbf{p}_j, \mathbf{p}_k)\}$. The robots should move so that the cost incurred is minimum of d_1 and d_2. Thus, we can define the cost of travel between two pairs as the minimum of d_1 and d_2. The cost metric defined above satisfies the triangle inequality. The TSP tour on the pairs using this cost metric will give the order in which the pairs of points should be traversed and hence also the assignment of points to the robots as they travel. An approach for solving the multidimensional TSP problem is discussed in [2].

C. Computational Results

We now provide computational results showing the performance of the greedy algorithm on example industrial data sets. However, the processing time assigned to each point is randomly generated in the range 0.01 to 0.1 seconds. We implemented the greedy algorithm in C++. The results of

TABLE I
PERFORMANCE COMPARISON OF GREEDY AND MATCHING ALGORITHMS FOR SPLITTING, WITH UNIT PROCESSING TIME FOR EACH POINT.

	Greedy Algorithm		Matching Algorithm		Ratio
Number of points	Processing Time (T_G) (sec)	Running Time (sec)	Processing Time (T_O) (sec)	Running Time (sec)	T_G/T_O
1396	788	1	700	5	1.13
11109	8537	8	6972	94	1.22
27810	16569	15	13905	1528	1.19

TABLE II

PERFORMANCE OF GREEDY ALGORITHM ON LARGE DATASETS.

Number of points	Processing Time (T_G) (sec)	Running Time (sec)	Sum of all Point Processing Times (T) (sec)	Ratio $2T_G/T$
1396	37.66	1	72.05	1.05
11109	407.47	8	554.82	1.47
27810	726.34	15	1390.99	1.04
135300	3435.78	25	6782.24	1.01
167536	4236.38	30	8388.49	1.01
198570	5009.33	40	9919.32	1.01
211856	5336.82	47	10564.50	1.01

the splitting problem using both the greedy algorithm and optimal (matching) algorithm along with the corresponding running times are shown in Table I. The value of the parameters used for obtaining the results are $\Delta = 8$ mm, $s_{\min} = 96$ mm. To compare our greedy algorithm results against the optimal results, we assumed all the processing times to be equal (one) and used an implementation of Edmond's algorithm available in the Boost Graph Library [10] to solve the MCM problem (which is equivalent to the MWM in this case). Table I compares the processing times of the greedy solution (second column) to that of the optimal solution (fourth column). It can be seen from Table I that the greedy algorithm is always within a factor of 1.5 of the optimal solution (see last column) and is much faster than the optimal algorithm (compare third and fifth columns). For the larger datasets, the MCM implementation does not return results within a reasonable time. So we have provided the greedy algorithm results for them. Table II shows the results for the greedy algorithm where the processing times of individual points vary from 0.01 to 0.1 seconds. The last column of Table II gives an upper bound on the ratio of the greedy cost to optimal cost (because the sum of all the point processing times is always less than or equal to twice the optimal cost). Table II (last column) shows that the greedy algorithm performs quite well for the large datasets and takes less than 1 minute (third column). The second dataset gives the worst result. However, in this point set there are at least 2835 points that have to be singletons even in the optimal solution (we can see this from the MCM solution). So the bound on the approximation ratio for this data set is not tight and the actual approximation ratio may be better. All the running times are obtained on an IBM T43p Thinkpad with a 1.8 GHz processor and 1 GB RAM.

IV. CONCLUSION

In this paper we considered the point assignment for processing problem for a constrained two-robot system covering a point set with non-identical point processing times. Since each robot can process one point at a time, the geometric (e.g., collision avoidance) constraints and kinematic constraints between the robots result in restrictions on points that can be simultaneously processed. We showed that the assignment problem can be solved optimally by converting it into a maximum weighted matching problem on a suitably defined graph. However, this $O(N^3)$ matching algorithm is too slow for large datasets and so we presented an $O(N^2)$ greedy algorithm and proved that it is guaranteed to return a solution within $3/2$ of the optimal solution. Note that our approach can handle fairly general geometric constraints between the robots, and is not restricted to planar point sets.

Future work includes computational comparison of our greedy algorithm solutions with approximation algorithm solutions for weighted matching, in terms of both solution quality and running time. The extension of the algorithms proposed here for $K > 2$ robots is also important. Relaxing the assumption that both robots are simultaneously stationary (when one or both robots are processing points) and obtaining algorithms that have constant factor approximation guarantees for the overall path planning problem (of point splitting and point ordering) is another important extension.

ACKNOWLEDGMENTS

This work is supported in part by the Center for Automation Technologies and Systems (CATS) under a block grant from the New York State Office of Science, Technology, and Academic Research (NYSTAR). John Wen was supported in part by the Outstanding Overseas Chinese Scholars Fund of Chinese Academy of Sciences (No. 2005-1-11). Srinivas Akella was supported in part by NSF under CAREER Award No. IIS-0093233.

REFERENCES

[1] B. Cao, G. I. Dodds, and G. W. Irwin. A practical approach to near time-optimal inspection-task-sequence planning for two cooperative industrial robot arms. *International Journal of Robotics Research*, 17(8):858–867, August 1998.

[2] N. Chakraborty, S. Akella, and J. T. Wen. Coverage of a planar point set with multiple constrained robots. In *IEEE International Conference on Automation Science and Engineering*, pages 899–904, Scottsdale, AZ, Sept. 2007.

[3] J. Edmonds. Maximum matching and a polyhedron with 0,1-vertices. *Journal of Research of the National Bureau of Standards*, 69B:125–130, 1965.

[4] A. B. Kahng, G. Robins, and E. Walkup. New results and algorithms for MCM substrate testing. In *Proc. IEEE Intl. Symp. on Circuits and Systems*, pages 1113–1116, San Diego, CA, May 1992.

[5] E. Lawler. *Combinatorial Optimization: Networks and Matroids*. Holt, Rinehert and Winston, New York, 1976.

[6] S. Micali and V. V. Vazirani. An $O(\sqrt{(|V|)}|E|)$ algorithm for finding maximum matching in general graphs. In *Proc. Twenty-first Annual Symposium on the Foundations of Computer Science*, pages 17–27, Long Beach, California, 1980.

[7] C. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice Hall Inc., Englewood Cliffs, NJ, 1982.

[8] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Prentice Hall, 2002.

[9] M. Saha, T. Roughgarden, J.-C. Latombe, and G. Sanchez-Ante. Planning tours of robotic arms among partitioned goals. *International Journal of Robotics Research*, 25(3):207–223, 2006.

[10] J. G. Siek, L.-Q. Lee, and A. Lumsdaine. *The Boost graph library: user guide and reference manual*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[11] I. Vasilescu, P. Corke, M. Dunbabin, K. Kotay, and D. Rus. Data collection, storage, and retrieval with an underwater sensor network. In *Proceedings of the International Conference on Embedded Networked Sensor Systems (ACM SenSys 2005)*, San Diego, CA, November 2005.

[12] D. E. D. Vinkemeier and S. Hougardy. A linear-time approximation algorithm for weighted matchings in graphs. *ACM Transactions on Algorithms*, 1(1):107–122, 2005.

[13] S.-Z. Yao, N.-C. Chou, C.-K. Cheng, and T. C. Hu. A multi-probe approach for MCM substrate testing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 13(1):110–121, January 1994.