4th IEEE Conference on Automation Science and Engineering
Key Bridge Marriott, Washington DC, USA
August 23-26, 2008

Distributed Multi-robot Work Load Partition In Manufacturing Automation

Girma S Tewolde, Changhua Wu, Yu Wang, Weihua Sheng

Abstract— **This paper presents strategies for systematic ways to deploy multiple mobile robots for servicing large numbers of points of interest in a distributed fashion. The mobile robots in such systems are responsible for providing several essential services. For example, in manufacturing automation, the tasks could include parts inspection, parts changing and data collection, etc. The efficiency, responsiveness, and service life of the mobile robots depend on the balanced allocation of the entire work load to the individual robots. Starting from an initial random deployment of the robots, the proposed distributed load balancing algorithm computes the load share of each robot using virtual potential force method. The load imbalance is used as the virtual force to dynamically move the robots until a more balanced distribution is achieved. The total travelling cost to visit all the points of interest in a partition and the cost of servicing individual points are combined to compute the load in each partition. To verify the effectiveness of the proposed algorithms, we conducted simulations by applying them in inspection applications and obtained satisfactory results.**

I. Introduction

A. Motivation

In various manufacturing and industrial scenarios, mobile robots are used to service a large number of points of interest [4], [6]. Mobile robots can be used to inspect and maintain large number of equipments whose locations cover a wide area. Each robot travels back and forth between the assigned equipments periodically to ensure the proper functioning of the equipments.

In the above mentioned scenarios, the cost of performing the assigned task has great effect on the outcome. Minimizing the cost will result in more efficient utilization of often limited resources and quicker response to urgent events. If the cost to service each point of interest is constant, then the total cost is dependent on the total travelling distance of the mobile robot, which is essentially a travelling sales person problem [1]. The mobile robot can travel along the optimal path or approximately optimal path. However, when there are multiple robots deployed at the same time, achieving optimal work load partition for each robot becomes a challenge.

In our previous work [5], we presented a distributed partition method for balancing the work load of multiple mobile robots. In that work, the initial positions of robots are randomly determined and the work load model only considers the cost at each point of interest. Therefore, the partition resulting from the proposed method depends much on the initial positions of the robots. The random initialization of the robot positions works well when the distribution of the points of interest are random. When the distribution of points of interest conforms to some mixture of multiple clusters, then it would be better to put the robots at the center or close to the clusters as demonstrated by Figure 1.

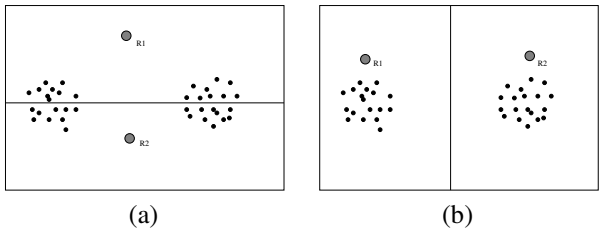


Fig. 1. The final partition of the work load depends largely on the initial position of the robots. Partition in (a) is obtained when the initial positions of the two robots are at the bottom and the top respectively. The partition in (b) is obtained when the two robots are initially positioned close to each cluster.

The previously proposed method works well for workload partitions in applications where the points of interest are closely distributed in the working environment and in cases where the cost of servicing the work requests is by far the dominant cost component compared to the travelling cost of the robots. However, in widely distributed environments, such as in large manufacturing plants, the travelling cost of the robots becomes a significant component of the total cost hence requiring special attention in the work load model. If the travelling cost is not taken into account, then the partitions may get to a state where robots have to travel long distances in order to service a few points of interest, as shown in Figure 2. In this paper, we propose a more realistic workload model which considers both the travelling cost and the cost of serving each point of interest. We also study the effects of the initial positions of the robots on the final load partition.

B. Problem Statement

We will consider the case where mobile robots are used for inspecting points of interest in a large environment. Let $X = \{x_1, x_2, ..., x_k\}$ be a set of points of interest and $C = \{c_1, c_2, c_3, ..., c_k\}$ be the *on-spot cost* that a robot has to spend for inspecting each point of interest. We assume that the robots do not have to go to the exact location of the points

Girma S Tewolde is with the Electrical & Computer Eng at Kettering University, Flint, MI 48504, USA gtewolde@kettering.edu
Changhua Wu is with the Department of Computer Science at Kettering University, Flint, MI 48504, USA cwu@kettering.edu
Yu Wang is with Department of Computer Science, University of North Carolina at Charlotte, USA. Email: yu.wang@uncc.edu
Weihua Sheng is with the School of Electrical and Computer Engineering at Oklahoma State University, Stillwater, OK, 74078, USA weihua.sheng@okstate.edu

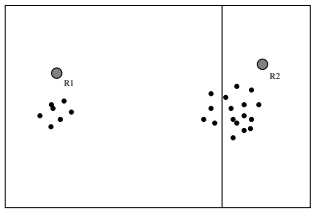


Fig. 2. If only the cost for servicing each point of interest is considered, then it may lead to a situation in which a robot has to travel long distances in order to service some points of interest, which demonstrates the need to take the travel distance into consideration during the load partition.

of interest for inspection. Range $H = \{h_1, h_2, h_3, ..., h_k\}$ defines the maximum distance a robot can be from the points of interest in order to perform its job. We can safely assume that the travelling cost $travel$ of the robots is linear to the travel distance. Therefore, the work load (cost) of a robot is the summation of its travelling cost $travel$ and the sum of all on-spot costs $service$ at points of interest it visits.

The robots have to periodically inspect the points of interest one by one. After finishing one round, the robots go back to their original positions. We want to find a work load partition such that each robot is only responsible for the points of interest in its partition and the work loads (cost) of all robots are approximately balanced or equal. Intuitively, such load balance strategy tends to extend the life time of the distributed system by minimizing the over-utilization of some of the robots and under-utilization of others. This is also expected to improve the response time of the robots to service requests.

In this paper, we propose a load partition algorithm which combines two load metrics. One is the travelling distance of the robot, and the other is the cost at servicing each point of interest. This paper is organized as follows. Section II presents the algorithm to calculate the optimized travelling cost for a mobile robot to service a partition. Section III presents a distributed approach for partitioning the work load among multiple mobile robots. Section IV presents our simulation experiments. Conclusion and future work are discussed in Section V.

II. Calculation of the Optimized Travelling Cost

In this section, we will briefly describe the algorithm for computing the travelling cost of servicing a set of points of interest using a mobile robot inside each partition. Here we only consider the set of points of interest in a partition, $X = \{x_1, x_2, \cdots, x_l\}$. As we have discussed in the problem statement, if we draw a circle around each point of interest x_i using its range h_i, then a robot has to get into the circle of a point of interest in order to service it. If the disks of multiple points of interest overlap, then the robot can simply go to the overlapping area and service all points of interest involved. We denote the areas formed by the circles and their intersections by $A = \{a_1, a_2, \cdots, a_l\}$. If an area a_i intersecting the circle defined by point of interest x_j, we call x_j can be covered by a_i.

There are two optimization problems in computing the travelling cost. The first optimization problem is how to select the minimum number of areas from A for the robot to visit to guarantee the coverage of all points of interest within a partition. This problem is actually the minimum set cover problem, which is a NP-hard problem [2]. We adopted a greedy heuristic for this problem. The greedy algorithm is described in Algorithm 1.

Algorithm 1 Greedy algorithm to select the minimum number of areas for covering the points of interest within a partition

Input: A set of areas $A = \{a_1, a_2, \cdots, a_l\}$ and a set of points of interest $X = \{x_1, x_2, \cdots, x_m\}$ within the partition.
Output: A subset of areas $A_S = \{a_{s_1}, a_{s_2}, \cdots, a_{s_n}\}$ for a robot to visit.

1: Initially, set all points of interest *uncovered* and the uncovered counter $k = m$. Let the potential coverage pc_i of each area a_i equal to the number of circles intersecting with this area.
2: **while** $k! = 0$ **do**
3: Select area a_j with the largest potential coverage pc_j (using IDs to break a tie) and add it into the selected subset A_S;
4: Mark all points of interest covered by a_j *covered*;
5: $k = k - pc_j$;
6: Update the pc_k for all adjacent areas a_k, by setting it to the number of intersected circles from *uncovered* points of interest.
7: **end while**

After selecting the area for a robot to visit, we need to decide their exact positions where a robot shall visit. Since the positions can affect the total length of the path that the robot needs to visit, we have to consider the position problem jointly with the path schedule problem. In our approach, we try to find a position in the selected area so that the total length of the path travelled by the robot is minimum. Our algorithm is an iterative algorithm in which in each step we add a new turn point inside one of the unvisited areas such that the distance added to the robot path is minimum. Remember that we have s_n areas to visit (output from Algorithm 1) and initially all areas are unvisited, our algorithm will terminate after s_n rounds, since in each round it adds a new turn point in the path and covers an unvisited area. Algorithm 2 shows the detailed algorithm.

Algorithm 2 is actually an approximation solution to travelling salesperson with neighborhoods (TSPN) problem [3], which is NP-hard. Notice that in Step 3 of Algorithm 2 we need to draw an ellipse which is tangent to a_j. This can be done by two ways. We can start with a small ellipse and increase its size until it reaches a_j. However, how to decide the initial size of the ellipse and what size to increase at each step are difficult. The second way is to use binary search. We first randomly select a point b inside a_j. We

Algorithm 2 Path schedule for a robot within a partition: to select the turn points of the robot

Input: A set of areas $A_S = \{a_{s_1}, a_{s_2}, \cdots, a_{s_n}\}$.
Output: A path $\Pi^D = v_0 v_1 v_2 \cdots v_n v_0$ which the robot shall visit.

1: Initially, set all selected areas a_{s_i} *unvisited* and the unvisited counter $k = s_n$. Let the path $\Pi^D = v_0 v_0$.
2: **while** $k! = 0$ **do**
3: For each edge on $v_i v_{i+1}$ in path Π^D and every unvisited area a_j, we draw an ellipse which uses v_i and v_{i+1} as its foci and is tangent to a_j. Let v_j be the tangent point. See Figure 3(a) for illustration. If select a_j to visit between v_i and v_{i+1}, the distance added to the path Π^D will be $||v_i v_j|| + ||v_j v_{i+1}|| - ||v_i v_{i+1}||$.
4: It is obvious that we want to select the unvisited area which adds the least distance to path Π^D. For example, in Figure 3(b), a_p, hence v_p, is a better choice than a_j. Assume we select a_p which is the best for all edges in Π^D and all unvisited areas, we mark a_p visited, and insert v_p between v_i and v_{i+1} in Π^D. Thus the number of edges in the path increases by one. $k = k - 1$.
5: **end while**

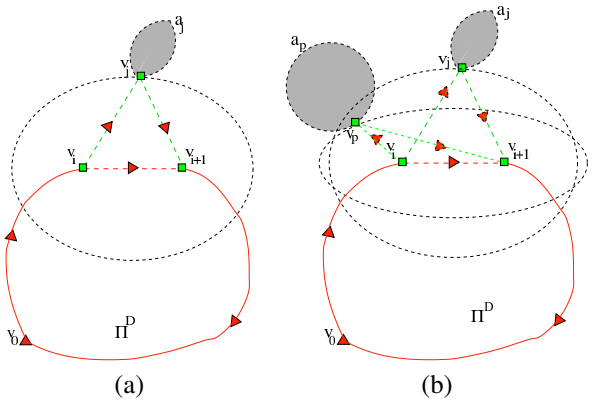


Fig. 3. (a) For each edge on $v_i v_{i+1}$ in path Π^D and every unvisited area a_j, we draw the ellipse which uses v_i and v_{i+1} as its foci and is tangent to a_j. The distance added to the path Π^D by visiting a_j is $||v_i v_j|| + ||v_j v_{i+1}|| - ||v_i v_{i+1}||$. (b) We select the unvisited area which adds the least distance to path Π^D. In this example, a_p is a better choice than a_j.

use $||v_i b|| + ||v_{i+1} b||$ as the major axis to draw the ellipse which guarantees to intersect with a_j. Then we reduce the major axis by half, if the ellipse does not intersect with a_j, we increase the major axis, otherwise further reduce it. By recursively doing this, we can find the ellipse which is tangent to a_j efficiently. In practice, if the range r_i is small compared with the distance between all areas, we can just use the ellipse via the center of a_i to estimate the optimal ellipse.

Figure 4 shows the path Π^D generated by Algorithm 2. Path Π^D represented by the line in the figure is the path that the robot will follow to service the points of interest, while the small squares are the turn points of the robot. In this example, ranges of all points of interest are set to be the same.

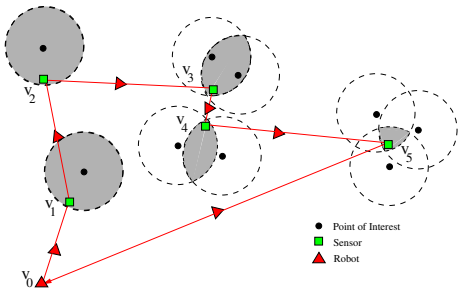


Fig. 4. Path Π^D generated by Algorithm 2. Here, the small squares are the turn points of the robot.

III. Distributed Multi-robot Load Partition Algorithm

In our previous work [5] a coordinated service set partition algorithm has been developed to divide a large working area into regions that each mobile robot is responsible for. The robots are in charge of providing the routine services to the points of interest in their partition. The goal of the partitioning strategy was to balance the workload in each partition. A simplified workload model was devised by considering the number of points of interest in the partition as the determining factor for the load. So the more points of interest there are the higher the load and vice versa. This model was easy to implement but it does not necessarily capture the actual work load in each partition.

A more realistic workload model is proposed here to capture the actual amount of energy and time the mobile robots would expend in their partition to carry out their mission. This proposed model considers not only the number of points of interest per partition but also their spatial distribution, as this affects the total distance the robot has to cover in order to provide the services to each of the points of interest in its partition. Hence the actual workload L_s in a partition s is computed as the total energy or time $travel_s$ the robot spends in travelling to all points of interest in its partition and the energy or time $service_s$ in servicing them. $service_s = \sum (c_i * x_i | x_i \in s)$, where c_i is the cost to service x_i in s as defined in the problem statement. Thus, $L_s = travel_s + service_s$

Algorithms 1 and 2 in the previous section present strategies for the coverage of distributed points of interest and for path planning to optimally cover them all. We employ these heuristic algorithms for computing the travelling cost component of the workload, $travel_s$. In the workload model considered in this paper we assume the cost of servicing the point of interest to be constant and the same for all points, i.e. $c_i = c$, for all i. The goal of the algorithm described below is to partition the total area of distributed points of interest into a given number of regions for each robot to take care so as to achieve a more balanced load distribution, according to the new workload model.

As in our previous service set partition (SSP) [5] algorithm, starting with initial partitions we employ the virtual

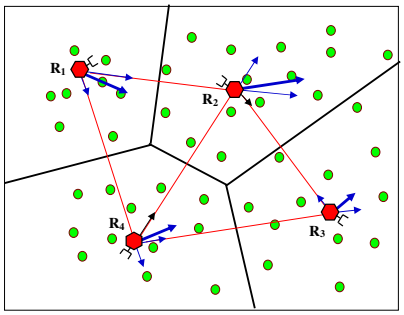


Fig. 5. Voronoi diagram for area partition and the virtual forces caused by load imbalances between neighboring partitions.

force method to iteratively adjust the partitions for each robot until the process leads to a more balanced load distribution. If a net non-zero virtual force acts on a robot, it will cause it to move in the direction of the force so as to modify the load distribution for a better balance. After the robots move in response to the applied forces the working area is repartitioned based on their new positions and the new load distribution is computed. This process is repeated iteratively until a balanced load distribution is achieved among all the robots.

However, it is very hard to achieve a perfect load balance between the partitions due to the discrete nature of the point of interest distribution. Initial attempts to run the algorithm for a perfectly balanced load distribution led to unstable or oscillatory performances. Hence to avoid instability behavior the goal of the algorithm is modified to a goal of minimizing the load differences between the different partitions. The performance metric used to measure the quality of the load partition is computed as the standard deviation of the load distribution from the average load. The detailed algorithm is given in Algorithm 3. With this approach the algorithm could be run until no more improvements are seen on the standard deviation of the load distribution for a pre-specified number of iterations.

When the knowledge of the cluster distribution of the points of interest is available, at the initial step of the load partition process the points of interest could be classified into separate clusters. For example, if we have q robots, then the points of interest can be classified into q clusters. So we can put the q robots at the centers of the q clusters. However, as shown in Figure 1, the initial position of the robots has influence on the final partition. The initial robot deployments and its associated Voronoi diagram result in initial partitions given by $P_1, P_2, \cdots, P_q$, where q is the number of partitions. Figure 5 illustrates the Voronoi diagram and the virtual forces attributed to the load imbalances.

IV. Experiments

To evaluate the performance of the multi-robot load partitioning algorithms with the aim of approximately balancing the loads in the different partitions we performed a series of simulation experiments as described in this section. Matlab is used in the simulation due to its computation and

Algorithm 3 Load balancing algorithm

1: Calculate the service load L_{P_i} of the subset S_{P_i}, for all $i = 1, 2, \cdots, q$.
2: Exchange service load with adjacent robots $R_{i1}, R_{i2}, \cdots, R_{ir_i}$, where r_i is the number of neighboring robots of robot R_i
3: Calculate the combined virtual force f_i on R_i

$$f_i = \sum_{j=1}^{r_i} \alpha(L_{sj} - L_{si}) \cdot \overrightarrow{n}_{ij}. \quad (1)$$

Here α is a scale factor and $\overrightarrow{n}_{ij}$ is the unit vector from R_i to R_j.
4: If $f_i \geq f_{th}$ then change the velocity of R_i according to the following equation

$$V_i = V_i + (f_i - \lambda V_i)/\beta \cdot \triangle t \quad (2)$$

Here f_{th} is a small threshold value, the term λv_i introduces some viscous force so that the deployment can be quickly stabilized, β is a constant for the robot mass, and $\triangle t$ is the time step size per iteration.
5: Using v_i update the new location of R_i as follows:

$$x_{R_i} = x_{R_i} + V_i \cdot \triangle t \quad (3)$$

6: Exchange new location information with neighbors and calculate the new Voronoi diagram and update the point of interest subsets in partition P_i
7: Check the stopping criteria, using the standard deviation of the load distributions among the different partitions as a quality metric
8: If the stopping criteria is met stop, else go to step 1

graphic capabilities. We consider several test configurations, with different dimensions of the working area and different number of points of interest randomly distributed in a large environment. Table I shows the list of the test configurations considered in the simulation experiments and the initial & final load distribution and the standard deviations of the corresponding load partitions. In each test scenario (except test cases 4 and 5), the points of interest are generated randomly at the start of the simulation. In all cases the locations of the points of interest are assumed to be known.

Based on the initial deployment positions of the robots the program sets out by first dividing the entire working area into different partitions (using Voronoi diagram, see Figure 5) and computing the initial load distribution in each partition. Then using the information about the initial load distribution the load balancing algorithm is executed to dynamically move the robots in the area in such a way that the difference in the load distribution among the different partitions is minimized. Virtual forces attributed to the load imbalances between neighboring partitions are responsible for the movement of the robots until equilibrium conditions are approximately achieved. Table I shows the results for the different test runs.

As can be observed from the results in the table (and visualized from Figure 8), the initial random deployment of

TABLE I

SIMULATION RESULTS OF THE LOAD BALANCING ALGORITHM

Test no.	Service area	n	ns	Initial load distribution		Final load distribution	
				Load in each partition	st. d.	Load in each partition	st. d.
1	40x40	4	40	37.1 26.2 121.8 84.1	44.2	73.6 77.0 79.3 71.6	3.4
2	40x40	4	40	46.3 126.7 64.9 29.2	42.5	79.4 74.2 78.5 77.4	2.4
3	40x40	4	40	28.0 80.6 59.2 101.1	31.2	63.1 62.8 61.8 67.4	2.5
4	40x40	4	40 (c)	41.8 33.7 89.1 48.2	24.7	37.0 44.1 42.7 40.7	3.1
5	40x40	4	60 (c)	42.2 59.7 63.7 42.8	11.2	57.2 49.3 57.1 57.8	4.0
6	40x40	5	40	60.5 18.2 31.4 36.4 70.7	21.6	55.5 59.5 57.0 55.8 60.0	2.1
7	40x40	5	40	69.2 56.9 18.1 88.7 36.1	27.6	55.0 51.4 48.3 50.8 51.0	2.4
8	40x40	5	40	33.1 21.2 107.8 38.0 26.8	35.5	56.2 51.1 46.8 52.8 50.5	3.4
9	50x50	5	50	78.0 36.2 126.4 48.0 31.4	39.3	87.1 86.1 82.6 80.5 94.4	5.3
10	50x50	5	50	101.1 79.8 47.9 60.1 56.1	21.4	84.2 82.2 76.5 80.8 76.5	3.5

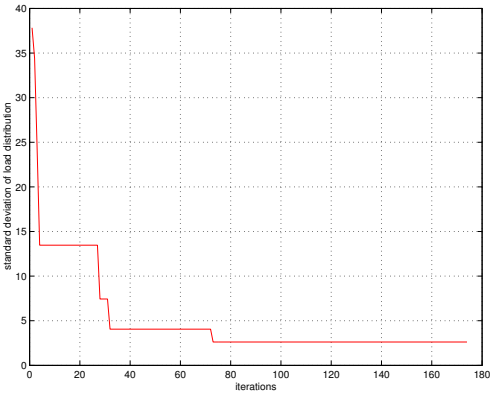
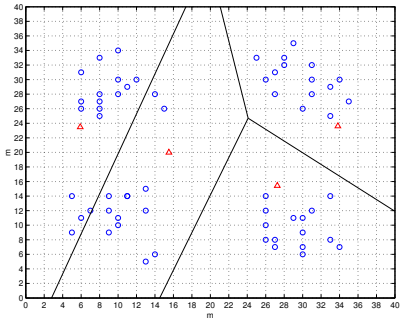
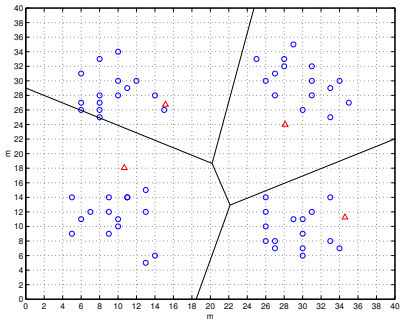


Fig. 6. Progress of the load-balancing algorithm.



(a) before load balance



(b) after load balance

Fig. 7. The effect of the initial position of robots on the load balancing, as applied to points of interest distributed in cluster formations (test 5).

the robots results in load distributions with large differences in the service load among the different partitions. This load imbalance is expressed quantitatively by the standard deviations of the loads carried by each robot. As the load balancing algorithm iteratively proceeds the load imbalance gradually decreases. The algorithm continues running until some termination condition is satisfied, which could be specified by either a minimum standard deviation or a maximum number of iterations without appreciable improvements. The standard deviation of the final load distribution indicates how closely the loads between the different partitions have been balanced. The graph in Figure 6 shows the progressive improvement of the standard deviation of the load distribution with increasing number of iterations.

To further understand how the distribution of the points of interest impacts the load balancing algorithm, we prepared two experimental configurations that contain points of interest distributed in clusters (experimental setups 4 and 5). The working areas for both setups are 40x40. In experimental setup 4 we generated 4 clusters with 10 points of interest randomly placed in each cluster, resulting in a total of 40 points of interest. In experimental setup 5, we also have 4 clusters but with 15 points of interest randomly distributed in each cluster, resulting in a total of 60 points of interest in the entire working area. The clusters are well separated spatially from each other. Figure 7 shows the test configuration in the experimental setup 5. The initial random placement of the mobile robots in the working areas results in non-optimal load distribution with large values of standard deviations. However, after the application of the load balancing algorithm, the resulting load distributions along the clearly defined clusters was produced with the expected smallest standard deviations.

V. CONCLUSIONS

This paper presented strategies for work load balancing of mobile robots in automated manufacturing environments, where a large number of points of interest need to be serviced. Starting with initial random deployment positions of the robots the algorithms were applied iteratively to redeploy the robots to positions that resulted in more balanced load distributions. A number of test runs have been performed

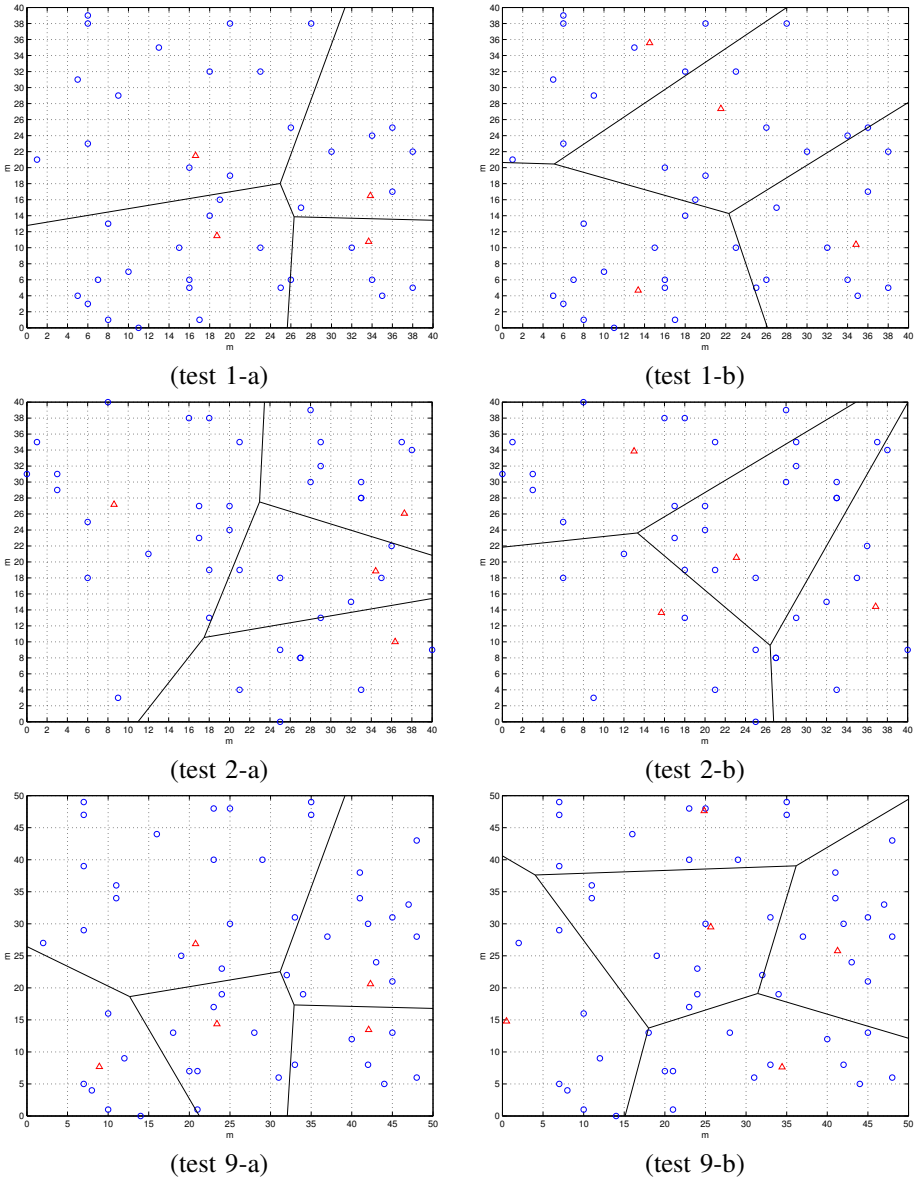


Fig. 8. Three test cases (test numbers 1, 2, & 9, in Table I). The left column shows the original partition. The right column shows the partition after load balance.

to verify the effectiveness of the algorithms. Future work will further investigate effects of clustering of the points of interest on the load balancing strategy and also closely study the relationship between the goals of minimizing load imbalances and that of minimizing total service load in the entire network. Since a minimum load imbalance does not necessarily lead to a minimum total service load this optimization goal requires techniques of combining the two minimization criteria.

REFERENCES

[1] D. Applegate, R. Bixby, V. Chavatal, and W. Cook. On the solution of traveling salesman problems. *Docuementa Mathematica –Extra Colume*, (3):645–656, 1998.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, , and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, Massachusetts, 2001.

[3] A. Dumitrescu and J. S. B. Mitchell. Approximation algorithms for TSP with neighborhoods in the plane. In *Symposium on Discrete Algorithms*, pages 38–46, 2001.

[4] Lars Kock Jensen, Bent Bruun Kristensen, and Yves Demazeau. Flip: Prototyping multi-robot systems. *Robotics and Autonomous Systems*, 53(3-4):230 – 243, 2005. Intelligent manufacturing;Agent architecture;Multi-robot systems;Autonomous transportation system;.

[5] W. Sheng and G.S. Tewolde. Robot workload distribution in active sensor networks. In *Proc. of 7th IEEE International Symposium on Computational Intelligence in Robotic and Automation*, Jacksonville, Florida, 2007.

[6] C. Son. Intelligent control planning strategies for mobile base robotic part macro- and micro-assembly tasks. *International Journal of Robotics and Automation*, 21(3):207 – 217, 2006. Macro and micro-assembly;Intelligent control planning;Fuzzy coordinator;Optimality criterion (fuzzy entropy);Machine intelligence;Vision sensors;.