Geometric Task Decomposition in a Multi-Agent Environment

Kaivan Kamali[1], Dan Ventura[2], Amulya Garga[3], and Soundar R.T. Kumara[4]

[1]Department of Computer Science & Engineering

[3]The Applied Research Laboratory

[4]Department of Industrial & Manufacturing Engineering

The Pennsylvania State University, University Park, PA 16802

[2]Computer Science Department

Brigham Young University, Provo, UT 84602

Abstract

Task decomposition in a multi-agent environment is often performed online. This paper proposes a method for sub-task allocation that can be performed before the agents are deployed, reducing the need for communication among agents during their mission. The proposed method uses a Voronoi diagram to partition the task-space among team members and includes two phases: static and dynamic. Static decomposition (performed in simulation before the start of the mission) repeatedly partitions the task-space by generating random diagrams and measuring the efficacy of the corresponding sub-task allocation. If necessary, dynamic decomposition (performed in simulation after the start of a mission) modifies the result of a static decomposition (i.e. in case of resource limitations for some agents). Empirical results are reported for the problem of surveillance of an arbitrary region by a team of agents.

1 Introduction

Multiagent teamwork is an active area of research and task decomposition among team members remains a challenge in multiagent environments. Some of the previous methods proposed for teamwork and task decomposition require periodic negotiations among team members (Cohen, Levesque, and Smith 1997; Cohen and Levesque 1991; Grosz 1996; Grosz and Kraus 1996; Levesque, Cohen, and Nunes 1990; Tambe 1997). Other methods require periods of limited and unlimited communication among team members (Stone and Veloso 1999; Stone and Veloso 2000). The methods mentioned above mainly perform task decomposition *after* the team starts the actual mission. This paper proposes a method for task decomposition with two phases that can be used *before* as well as *after* the start of the actual mission. The two phases of the proposed method are called *static* and *dynamic* decomposition respectively and are described below.

1. Static decomposition: The task is decomposed among team members using computational geometric techniques. The team then starts the mission (in the simulation) and the performance of the team is measured. This process is repeated many times in simulation and the decomposition yielding the best team performance is chosen and is actually implemented by the team in the real mission.

2. Dynamic decomposition: It is assumed that the task is already decomposed among team members by static decomposition. The decomposition is modified to improve the team performance if the team is not able to finish the overall task (e.g. some agent runs short of power)

The problem addressed in this paper is the surveillance of an arbitrary region by a team of agents. The Voronoi diagram (VD) (de Berg, van Kreveld, Overmars, and Schwarzkopf 2000; Okabe, Boots, Sugihara, and Chiu 2000) is used for decomposition of the region among team members, and software tools were developed to perform the experiments.

The rest of the paper is organized as follows: Section 2 briefly discusses the Voronoi diagram, its applications, and algorithms for its computation. Section 3 explains the methodology and why the VD is used for the task decomposition problem. Section 4 describes the proposed algorithm for the VD calculation. Experimental results are given in Section 5 and in Section 6 the work is summarized and future work is discussed.

2 Voronoi Diagram (VD)

The VD is a geometric structure that gives proximity information about a set of sites. Given a set of sites in a 2-dimensional plane, the VD partitions the plane into cells so that any point within a cell is closest to the site in that cell. The points that are equidistant from two or more sites form the border of the cells. The following notation is introduced before giving the mathematical definition of the *planar ordinary Voronoi diagram* (Okabe, Boots, Sugihara, and Chiu 2000):

- $\Re^2$: Euclidian plane
- n: number of points in $\Re^2$
- π: point i in $\Re^2$
- $x_i = (x_{i1}, x_{i2})$: Cartesian coordinates of π

2.1 Definition a Planar Ordinary VD

Let $P = \{p_1, \ldots, p_n\} \subset \Re^2$, where $2 < n < \infty$, and $x_i \neq x_j$ for $i \neq j$, and $i, j \in I_n = \{1, \ldots, n\}$. The region given by $V(p_i) = \{x | \|x - x_i\| \leq \|x - x_j\|$ for $j \neq i$, and $i, j \in I_n = \{1, \ldots, n\}\}$ is called the *planar ordinary Voronoi polygon* associated with p_i and the set given by $V = \{V(p_1), \ldots, V(p_n)\}$ is called the *planar ordinary Voronoi diagram* generated by P. We call p_i of $V(p_i)$ the generator point of i^{th} Voronoi polygon, and the set $P = p_i, \ldots, p_n$ the *generator set* of the Voronoi diagram V. In the literature and in this paper, a generator point is sometimes referred to as a *site*. Voronoi diagrams have been applied in many different areas including telecommunication, biology, ecology, medicine, path planning, and geophysics. VD is used to model a communication network in (Baccelli, Klein, Lebourges, and Zuyev 1997). In (Luchnikov, Medvedev, and Gavrilova 2001) it is used to develop software that simulates a biological molecule. VD has also been used for the design of neural networks (Bose and Garga 1993). Other applications of VD include color segmentation of medical images (Imielinska, Downes, and Yuan. 2000), robotic path planning (Allen, Stamos, Gueorguiev, Gold, and Blaer 2001), and interpolation of geologic properties, such as temperature (Sambridge, Braun, and McQueen 1995).

There are several algorithms for the VD computation. These include the *half-plane intersection algorithm* (de Berg, van Kreveld, Overmars, and Schwarzkopf 2000), *Fortune's algorithm* (de Berg, van Kreveld, Overmars, and Schwarzkopf 2000), and the construction of VD through its dual, the Delaunay tessellation, using neural networks (Garga and Bose 1994). The half-plane intersection algorithm calculates the VD by finding the common intersection of the half-planes generated by the bisectors of the lines connecting each site to all other sites. The half-plane intersection algorithm presented in (de Berg, van Kreveld, Overmars, and Schwarzkopf 2000) spends $O(nlogn)$ time to compute each Voronoi cell, leading to a total time of $O(n^2 logn)$ for n sites. The most efficient algorithm for the VD computation is the Fortune's algorithm (de Berg, van Kreveld, Overmars, and Schwarzkopf 2000), a plane-sweep algorithm, which computes the VD in $O(nlogn)$.

3 Problem Description

The following notation is introduced before discussing the problem description:

- n: number of agents in the team
- A: area to be surveyed by the team
- a_i: agent i, $1 \leq i \leq n$
- e_i: resource (power) of agent a_i
- c_i: Voronoi cell assigned to a_i
- p_i: site location for c_i

- S_i: left-most (lower-most) point of c_i, (starting point for a_i)
- g_i: time for a_i to get to S_i
- s_i: time for a_i to survey c_i
- t_i: g_i + s_i , total time for the agent to finish the assigned task
- $P(D)$: team performance, $max(t_i)$, $1 \leq i \leq n$, for decomposition D of region A

A team of agents is to survey an arbitrary region. Each agent has a finite amount of power associated with it. The problem to be solved is the allocation of the region among team members, and it is to be solved for two different cases:

1. All the agents have enough power to survey the area allocated to them.
2. Some of the agents do not have enough power to survey the area allocated to them.

In the first case, it is desired to do the allocation in such a way that the team can finish the overall surveillance in the shortest amount of time. In the second case, it is desired to modify the allocation such that the team can survey the whole area. If this is not possible, then the allocation should increase the percentage of the overall area being surveyed, while making sure no other agent runs short of power and that the power shortage remains restricted to the initial agents with power shortage.

Voronoi diagrams have been applied to many different areas from path planning to communications networks. Here the VD is used for sub-task allocation among team members. Some of the properties of the Voronoi diagram that make it suitable for the task decomposition problem are the following:

1. Computation of the Voronoi diagram is very efficient, especially for 2-dimensional spaces.
2. Small changes in the site locations will produce only local changes in the VD. Hence if the changes to the site locations are small, the new VD can be computed efficiently (Garga and Bose 1994).
3. Voronoi diagrams present a compact representation of the task decomposition problem.

Given some site locations within an arbitrary region, the VD splits the region into cells with one site residing in each cell. The resultant cells have the property that any point in them is closest to the site in that cell. The points that are equidistant from two or more sites form the borders of the cells. The VD is generally used for problems in which the site locations are fixed (e.g. the famous Post Office problem (de Berg, van Kreveld, Overmars, and Schwarzkopf 2000)). For the task decomposition problem addressed in this paper the site locations are also fixed (even though the location of the agents can vary). Note that the location of an agent assigned to a cell is unrelated to the site location use to generate that cell. The

```
Input: the area to be surveyed ($A$), and the number of agents ($n$)
Output: decomposition of area $A$ among $n$ agents, where agent $a_i$ surveys Voronoi cell $c_i$, $\forall\ 1 \leq i \leq n$
Repeat
    $L = p_i, \ldots, p_n$    //Select $n$ random site location in area $A$
    $D$ = Voronoi($L$)    //Compute the Voronoi diagram based on site locations in $L$
    Survey($D$)    //Agent $a_i$ surveys Voronoi cell $c_i$ and finishes surveillence in time $t_i$, $\forall\ 1 \leq i \leq n$
    $P(D)$ = max($t_i$), $1 \leq i \leq n$    //The performance of the team is determined
Until (stopping condition true)
```

Table 1. Static decomposition algorithm

location of an agent assigned to a cell varies as the agent moves and can be anywhere in the overall region, while the site location of a cell is fixed and is just used, initially, to calculate the VD. Since the agents have limited power, they will try to conserve power by not moving around while the decomposition is being performed. Random locations are chosen as site locations and the corresponding VD is calculated. The agents perform the surveillance in simulation based on the resulting VD. This process is repeated in simulation many times, and the performance of the team for different decompositions is compared. The performance of the decomposition is then compared with a baseline decomposition to determine whether the VD is a good method for task decomposition for this problem. The static decomposition algorithm is given in Table 1.

We used a maximum number of iterations as our stopping criteria for the static decomposition algorithm. This criteria is a standard approach in Genetic algorithms and in many variations of gradient descent and we showed that the criteria produces acceptable results (Section 5). The users can determine the maximum number of iterations based on their priorities (what is acceptable to them and their time constraints). It should be noted that we are not looking for the optimal decomposition – we are looking for a good solution, something that is ”approximately” optimal. We generate several random decompositions and take the best one, where best is determined based on the use of resource (in this case the resource is time). We obtained empirical results (given in Section 5) that show that the proposed method produces good results and does so with limited computational overhead. The goal of the paper is not on finding the optimal decomposition or on finding it in the most computationally efficient way. The goal is a proof-of-concept – VD can be applied to task decomposition among a team of agents. The methods work well empirically and can achieve good results in conserving resources considered critical with minimal computational overhead.

3.1 First Case

In the first case of the problem, the performance of the team is measured based on how long it takes for the team to finish the overall surveillance. After repeating the process many times in simulation, the best allocation is determined by comparing time performance of different decompositions (i.e. different VDs). It is assumed that all the agents begin their surveillance from their starting point (S_i). If the initial location of the agent is not the same as S_i, then the agent must first go there before starting the surveillance of the cell and the cost associated with the agent going there must also be considered. Figures 1 and

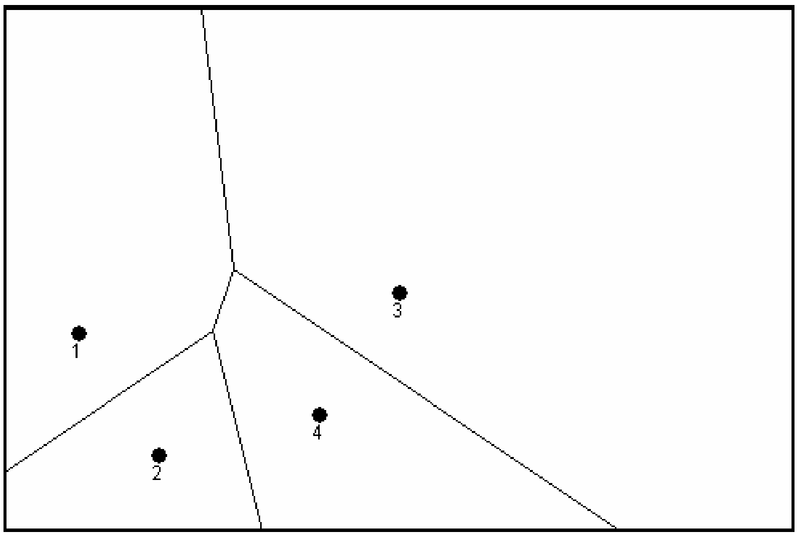

Figure 1. Example of inefficient task decomposition

2 show two different VD generated by random site locations. It is assumed that agents start at the S_i corresponding to their cells (this constraint will be relaxed later) and there is no additional cost associated with the agents getting there. Each agent is tasked with surveying the cell assigned to it.

As shown in Figure 1, random site locations can lead to poor task decomposition. Agent number 2 has a much larger cell to survey than the other agents, and since the agents are working in parallel, the completion of the overall task is delayed by agent number 2 surveying the large cell assigned to it. In Figure 2 the random site locations divide the region among agents more evenly and the performance of the team is improved since all the team members finish their tasks relatively close to each other in time.

It is important to note that the above conclusions are based on the assumption that each agent is initially located at the S_i of its corresponding cell. If this assumption is relaxed, nothing can be said about the relative performance of the above two allocations – the performance depends on the initial location of the agents. For a certain initial location of the agents, the first allocation might lead to a better team performance than the second.

4 Second Case

In the second case of the problem, with the power constraints, the goal is to come up with an allocation such that all the agents can finish the surveillance of their assigned cells. If some agents do not have enough power to finish the surveillance of their assigned cells, the overall task must be reallocated so that they are assigned smaller cells. We will first describe the dynamic decomposition algorithm in Table 2, and then will demonstrate the solution for the single and multi-agent cases.

The site location of a cell whose agent has surplus power can move closer to the site location of the agents with power shortage up to the point where the agent with surplus power would run short of power itself if given any larger cell assignment.

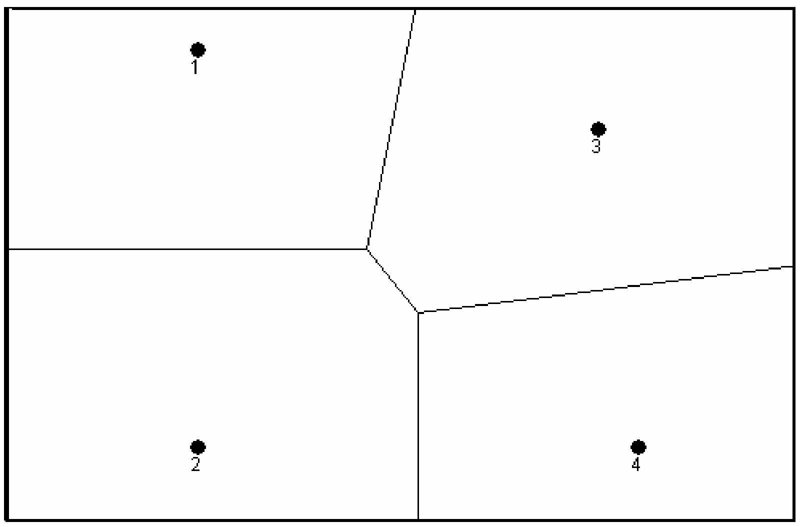

Figure 2. Example of efficient task decomposition

```
Input: $p_i$, $e_i$, and $\Delta$, $\forall\ 1 \le i \le n$      //sites, resources, movement scaling factor
Output: $p_i$, $\forall\ 1 \le i \le n$                     //modified sites
Do
    For i = 1 to $n$
        If $e_i > 0$    //for each agent with extra resources
            For $j$=1 to $n$ and $j \ne i$
                If $e_j < 0$    //for each agent lacking resources, move the site
                                //locations agents with extra power closer
                    $p_i(x) = p_i(x) + \Delta * (p_j(x) - p_i(x))$    //movement of x coordinate
                    $p_i(y) = p_i(y) + \Delta * (p_j(y) - p_i(y))$    //movement of y coordinate
Until(($e_i \ge 0\ \forall i$) or ($e_i \le 0\ \forall i$))    //until all agents have enough resources or
                                                   //all agents have no extra resources
```

Table 2. Dynamic decomposition algorithm

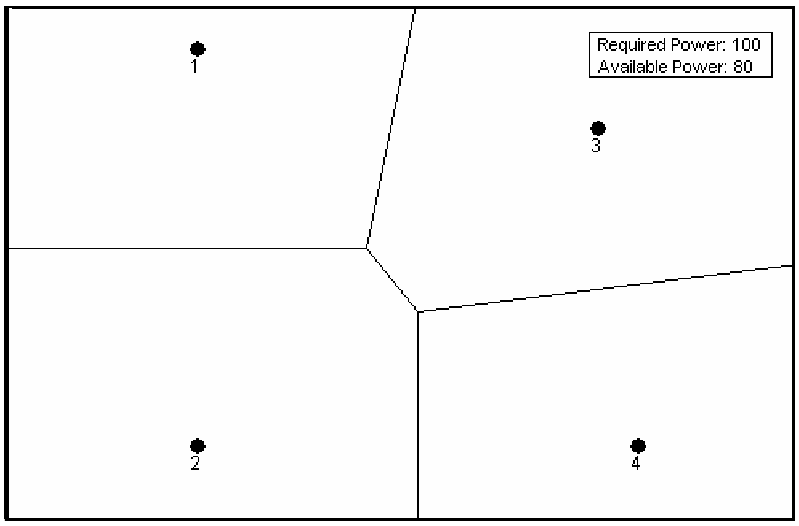

Figure 3. The cell assigned to agent 3 (surveying cell 3) is too big relative to the its power

Further movement of the site location would result in another agent with power shortage. If the dynamic decomposition algorithm stops in step 2, the agent with power shortage has succeeded in surveying the newly assigned (and smaller) cell without any other agent running short of power and the goal is achieved. If the dynamic decomposition algorithm stops in step 1, then it is impossible to do the overall task, but the area surveyed is significantly increased while no other agent suffers from power shortage.

In Figure 3, agent number 3 does not have enough power to finish the assigned task because the cell assigned to it is too big relative to its power reserves.

The site locations of the cells whose agents have enough power are moved closer to the site location of the cell whose agent is lacking enough power, and the VD is re-calculated. The agent lacking power will now have a smaller cell assigned to it since other site locations have moved closer and the cell borders are pushed back. Figure 4 shows the movement of the site locations and the resulting modified VD. It is clear that agent number 3 now has a smaller cell to survey.

In Figure 5 agents 1 and 2 have surplus power and agents 3 and 4 are short of power. In the case of multiple agents having power shortage, the displacement of the site location of agents with surplus power is the vector sum of the displacement of the site location toward each agent with power shortage. The site locations of agents 1 and 2 are moved and the new site locations are determined based on the vector sum of displacement toward the site locations of agents 3 and 4. Hence agents 3 and 4 will have smaller cells to survey. The new Voronoi cells are drawn with bold lines.

One nice feature of the algorithm is that it is iterative and can dynamically adapt to further changes in the environment (e.g. another agent runs short of power). It is also optimal in the sense that small changes in site locations will produce only local changes to the overall VD and hence will reduce the computational cost substantially. In the dynamic decomposition algorithm, the site location of the agents with surplus power is moved toward the site location of the agents with power

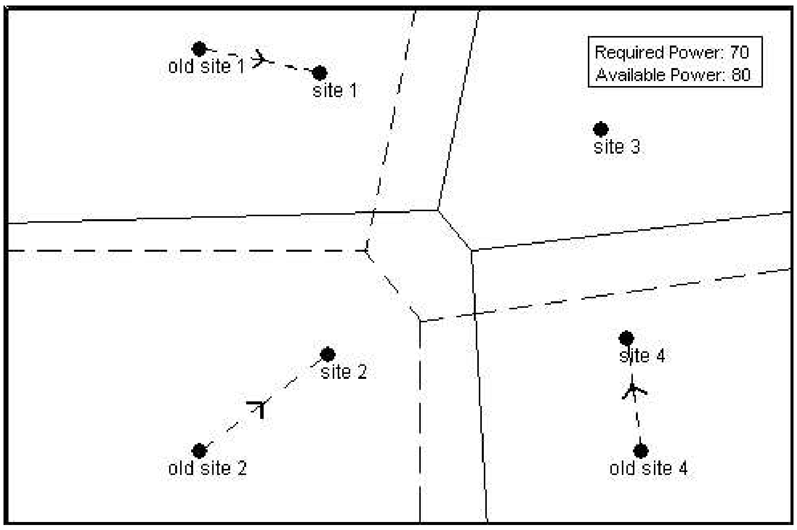

Figure 4. The site location of cells whose agents have surplus power are moved closer to the site location of the cell whose agent has deficient power and the VD is re-calculated. The old Voronoi cells are shown with dashed lines

shortage in small increments, repeatedly, so that the agents with surplus power do not run short of power themselves. To make the site movements more efficient and procedural, the increments in which the site locations are moved are determined based on the following criteria:

1. The distance between the site location of the agent with surplus power and the site location of the agents with power shortage, d_i. In case of multiple agents with power shortage, this would be a vector sum of the vectors connecting the site location of agent with surplus power to the site location of all the agents with power shortage.
2. The amount of surplus power the agent has, e_i.

The site location of agents that are closer to the agent with power shortage will move in larger increments than the agents whose site locations are farther. Also the agents with more surplus power will move their site locations in larger increments. The first condition will localize the effect of the dynamic decomposition, with the neighboring agents carrying most of the burden. The second condition makes the agents with more surplus power contribute more than those with less power. To achieve the abovementioned effects, the movement scaling factor in the dynamic decomposition algorithm, Δ , is defined as $\Delta = (e_i/e_{max} * (1 - d_i/d_{max}))$, where e_{max} and d_{max} normalize e_i and d_i respectively. e_{max} can be thought of as the maximum surplus power an agent can have and d_{max} as the maximum dimension of the area that the agents are working in.

Another possibility for moving the site locations is to use an appropriate optimization technique such as genetic algorithms (GA). A population is created based on different site locations and the fitness of each chromosome is determined based on the performance of the team for that specific decomposition. Genetic operators can be applied to the population to guide the search.

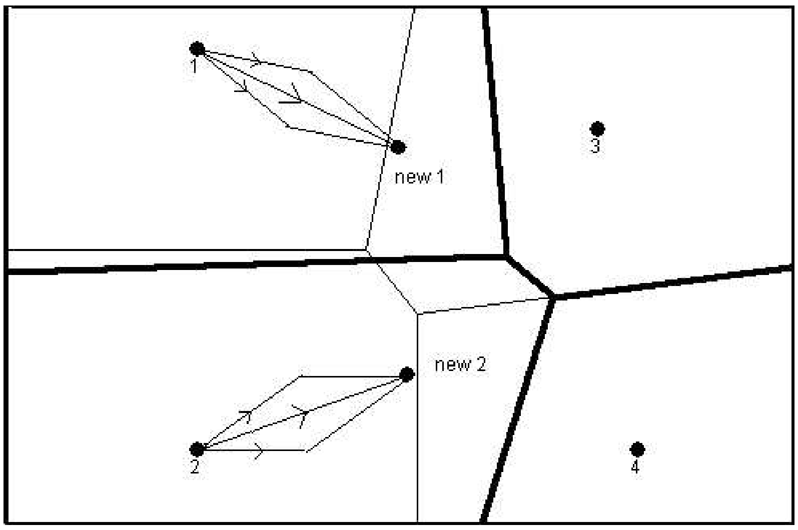

Figure 5. The task is reallocated for the case of agents 3 and 4 having power shortage. The new Voronoi diagram is drawn with bold lines

5 Proposed Voronoi Algorithm

The idea behind the proposed algorithm is similar to that of the half-plane intersection algorithm (de Berg, van Kreveld, Overmars, and Schwarzkopf 2000) but intersection of half-planes is done in such a way that additional information is obtained about vertices and edges of the cells. The additional information is necessary to describe the Voronoi cells to the agents that are assigned to them. The complexity of the proposed algorithm is more than the complexity of the half-plane intersection algorithm, but the half-plane intersection algorithm does not provide the necessary additional information explicitly and obtaining such information will increase its complexity. The algorithm is described in Table 3.

5.1 Complexity Analysis

If there are n sites, then there will be $(n-1)$ bisectors for the lines connecting each site to all other sites. Each bisector of a site will intersect other bisectors of that site in at most $(n-2)$ points. Each of those points must be checked to see whether it is a valid point on the border of the Voronoi cell. If the point satisfies the inequalities imposed by the $(n-1)$ bisector equations, then it is a valid point on the border of the Voronoi cell. Since there are n sites, $n-1$ bisectors for each site, $n-2$ points on each bisector of each site, that need to be checked against $(n-1)$ inequalities, the overall complexity of the algorithm is $O(n^4)$. This is more than the complexity of the half-plane intersection algorithm, $O(n^2logn)$, but as mentioned before, the half-plane intersection algorithm does not provide the necessary additional information explicitly and obtaining such information will increase its complexity.

```
Input: { $p_i$ }, $\forall\ 1 \leq i \leq n$          // sites
Output: { $c_i$ }, $\forall\ 1 \leq i \leq n$         // Voronoi cells
For i = 1 to $n$                             // for each site compute the bisector of the line
    For j = 1 to $n$ and $j \neq i$              // connecting that site to all other sites
        $B_{ij}$ = computeBisector($p_i$, $p_j$);

For i = 1 to $n$                             // for each site, and for the bisectors of each site,
    For $j$ = 1 to $n$ and $j \neq i$              // calculate the intersection of each bisector with
        For $k$ =1 to $n$ and $k \neq j$           // all other bisectors of the same site and store
            $I_{ijk}$ = computeIntersections($B_{ij}$,$B_{ik}$)        //them in $I_{ijk}$

For i = 1 to $n$                             // determine elements of Iijk that are valid points
    For $j$ = 1 to $n$ and $j \neq i$              // on the border of $c_i$, save them in $V$
        For $k$ =1 to $n$ and $k \neq j$           // A point is valid if it satisfies all the inequalities
            For $m$ = 1 to $n$ and $m \neq i$       // imposed by all the bisector equations
                $V_{ij}$ = satisfiesInequality($I_{ijk}$, $B_{im}$)

For i = 1 to $n$                             // for each site, $c_i$ borders are determined based
    For $j$ = 1 to $n$ and $j \neq i$              // on the valid points on each bisector
        $CB_{ij}$ = getCellBorders($V_{ij}$)
    $c_i$ = constructCell({ $CB_{ij}$ } )             // ci is known given all the cell borders
```

Table 3. The proposed Voronoi diagram algorithm

6 Experimental Results

Three sets of experiments were performed using agents in a simulated environment and are discussed next.

6.1 Static Decomposition

The first set of experiments compares the performance of static decomposition with a base-line decomposition. A team of n agents is assigned to a rectangular region A (in general the shape of A can be arbitrary; rectangular shape was chosen for ease of illustration). The area is decomposed among team members by computing the VD of A (with site locations generated randomly) and assigning a_i to c_i. It is assumed that a_i is initially at S_i. Therefore g_i = 0 and t_i = s_i. This assumption will be relaxed for the second set of experiments. Then a_i surveys c_i using a lawnmower-patterned search, scanning alternatively from top to bottom (then from bottom to top) while moving from left to right across the cell (Figure 6). Each agent has a survey radius r and as long as the movement to the right is smaller than $2r$, it is guaranteed that the whole area will be covered. It is assumed that the speed at which the agents do the surveillance is fixed for all agents, so that the size of a cell is directly correlated with the time s_i required to survey it.

The goal in the first experiment is to minimize $P(D)$ and to compare it with the baseline decomposition, which for the special case of g_i = 0 is known. The team is finished with the overall task when all the agents are done surveying the cells assigned to them. Since the agents perform their tasks in parallel, performance of the team is determined by the agent that

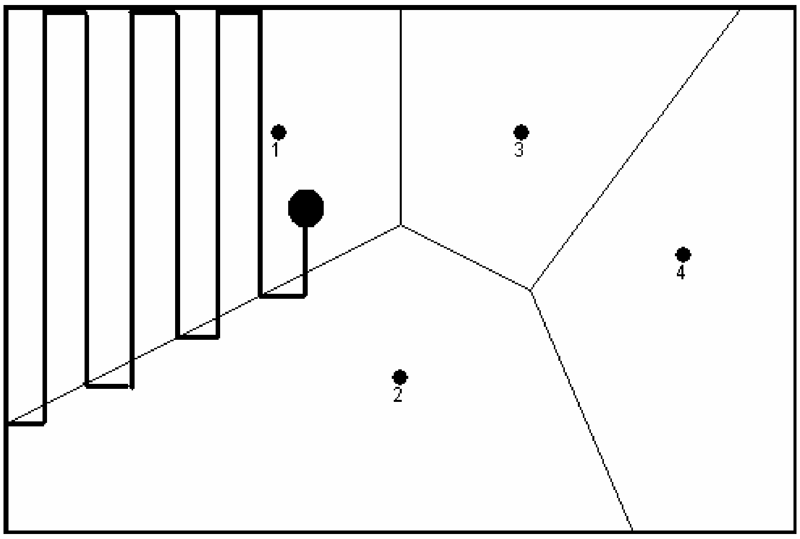

Figure 6. Agent 1 is surveying cell 1, to which it is assigned. It will survey the cell based on a lawnmower-patterned search, scanning alternatively from top to bottom (then from bottom to top) while moving from left to right across the cell. The agent starts the search from the leftmost (lower-most) point of the cell

finishes the last in time, i.e. the agent with the largest value for t_i.

After the team is finished with the overall task, the largest value for t_i, $1 \leq i \leq n$, is selected and represents the performance of the team for D, the decomposition of the region. The notation $P(D) = max(t_i)$ is used to represent the performance of the team for decomposition D.

The baseline decomposition divides the area evenly among team members (this decomposition is denoted by O), resulting in all the agents finishing their sub-tasks in about the same time. If the tasks were not divided evenly, the agent that was assigned to the largest of the cells would keep the rest of the team waiting, degrading the performance of the team. Therefore, in order to have a metric for comparison, given n agents, the region was divided into n equal-sized cells. Table 4 shows the results obtained from running the experiments. The experiments were performed for teams of 2, 3, 4, 5 and 6 agents. We used a maximum number of iterations as our stopping criteria for the static decomposition algorithm. The decomposition of the region was done 50, 100, 200 and 400 times by computation of random VD. The best, the worst, the median, the average and the standard deviation of the team performance are shown in Table 4. The best team performance for all teams and for every experiment is graphed in Figure 7. The results show that the best team performance compares favorably with the baseline solution. Also, apart from minor fluctuations, the best team performance is improved as the number of decompositions becomes larger.

n, Number Of Agents ⇩	*k*, Number Of Iterations ⇨	**50**	**100**	**200**	**400**	*P*(*O*)
2	**Best**	76118	75921	76067	75883	75447
	Worst	138237	144907	147722	150927	
	Median	99292	98862	96513	97041	
	Mean	99779	100845	99913	100683	
	Std. Dev.	17083	16581	17428	17042	
3	**Best**	55185	53912	52588	51487	51397
	Worst	129466	126579	129452	128905	
	Median	70482	74945	75729	73044	
	Mean	76100	77609	77894	76404	
	Std. Dev.	15348	16222	15075	15827	
4	**Best**	42238	42143	40705	39234	38922
	Worst	101334	114563	122465	112782	
	Median	60730	58474	59878	60732	
	Mean	62924	61534	62068	63457	
	Std. Dev.	13124	13946	12630	14353	
5	**Best.**	36406	34125	33985	33445	31437
	Worst	84049	93691	100414	102832	
	Median	51569	51672	50222	52110	
	Mean	52503	54078	52863	54239	
	Std. Dev.	11212	12512	11501	12447	
6	**Best**	31589	28762	30368	28408	26447
	Worst	89010	80225	76614	85782	
	Median	45791	43528	44796	45230	
	Mean	47605	44725	46482	47088	
	Std. Dev.	11377	9331	10267	10394	

Table 4. The time (in arbitrary units) required for n agents to survey a rectangular area A, based on decompositions by k random VD computations. The rightmost column is the team performance for the benchmark solution

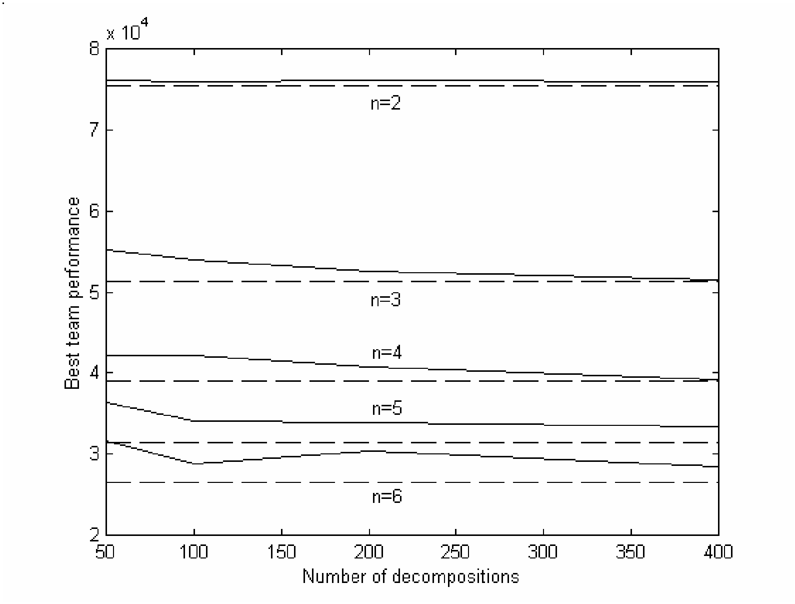

Figure 7. The best team performance for all teams and all task decompositions is depicted. The dashed-lines represent the team performance for base-line solution

6.2 Static Decomposition (Revisited)

In the first experiment it was shown that the proposed task decomposition method compares favorably with the base-line decomposition. The base-line decomposition was known and was used as a metric for evaluating the performance of the proposed method. In the second experiment, it is assumed that a_i is not initially at S_i. So in order to start the survey, a_i must first go to S_i, which is the starting point for the search. The performance of the team is determined by the time that it takes for the agent to get to the starting point, g_i, plus the time that it takes for the agent to survey the assigned cell, s_i, i.e. $t_i = s_i + g_i$.

When $g_i \neq 0$, dividing the region into equal-sized cells will not in general lead to the best solution, as g_i is ignored. Unequal partitioning of the region will, in the general case, lead to better team performance, i.e. the total cost ($t_i = s_i + g_i$) will be less than the total cost for equal-sized partitioning of the region. In the second experiment, the region is repeatedly allocated among agents by computing the VD of the region (with site locations generated randomly) and the team performance is measured. The VD is computed 50, 100, 200 and 400 times. Since there is no known optimal solution, in order to have a metric for comparison, the team performance is compared with the performance of a team assigned to a group of equal-sized horizontal (and vertical) cells. These decompositions are called H and V respectively. Even though these are not the optimal solutions, they are an intuitive and natural decomposition and provide a metric for evaluation of the method used. Table 5 shows the results obtained from running the experiments. The best team performance for all teams and for every experiment is graphed in Figure 8. Some fluctuations are noticed in the graph but generally the team performance improves as the number

n, Number Of Agents	*k*, Number Of Iterations	**50**	**100**	**200**	**400**	***P*(*H*)**	***P*(*V*)**
2	**Best**	87773	84374	84646	83095	90947	90047
	Worst	150787	157432	160247	163727		
	Median	111163	109859	108049	108985		
	Mean	111856	112050	111328	112168		
	Std. Dev.	17135	17042	19106	17695		
3	**Best**	62646	65867	62591	59874	65997	64898
	Worst	143091	141204	142764	140015		
	Median	79851	83752	85133	82335		
	Mean	85463	87413	87211	85685		
	Std. Dev.	15930	16290	15902	16454		
4	**Best**	49860	49849	49886	46787	53522	52172
	Worst	110039	126134	132834	123565		
	Median	73739	72034	69671	69584		
	Mean	74473	73402	71930	72142		
	Std. Dev.	12240	14003	13977	14683		
5	**Best**	44543	43138	41612	40460	46037	44597
	Worst	113407	110459	110822	118595		
	Median	60163	60587	59481	60383		
	Mean	62180	63879	61936	62505		
	Std. Dev.	11681	12705	11665	13246		
6	**Best**	37262	35463	38159	37576	41047	39749
	Worst	84259	86628	105090	100004		
	Median	55084	52558	52264	53233		
	Mean	55620	54204	55312	55462		
	Std. Dev.	10454	9851	11781	11139		

Table 5. The time (in arbitrary units) required for n agents to survey a rectangular area A when the agents' initial locations are not correlated with the locations of their assigned cells, based on decompositions by k random VD computations. The rightmost columns are the performance of the team for a group of horizontal and vertical, equal-sized cells, used as a metric for comparison with the results of the proposed method

of random decompositions is increased. The important point to be noted is that the decomposition yields a team performance that is better than either of the two intuitive cases considered for comparison.

6.3 Dynamic Decomposition

The third experiment assesses the performance of dynamic decomposition. It is assumed that the task has been initially decomposed among agents via static decomposition but one of the agents does not have enough power to finish the assigned sub-task. The task decomposition is modified by dynamically modifying the VD. The site location of agents with surplus power are moved incrementally closer to the site location of agents with power shortage and the VD is re-calculated, thus reducing the size of the Voronoi cell assigned to the agents with power shortage. In simulation the team performs the task

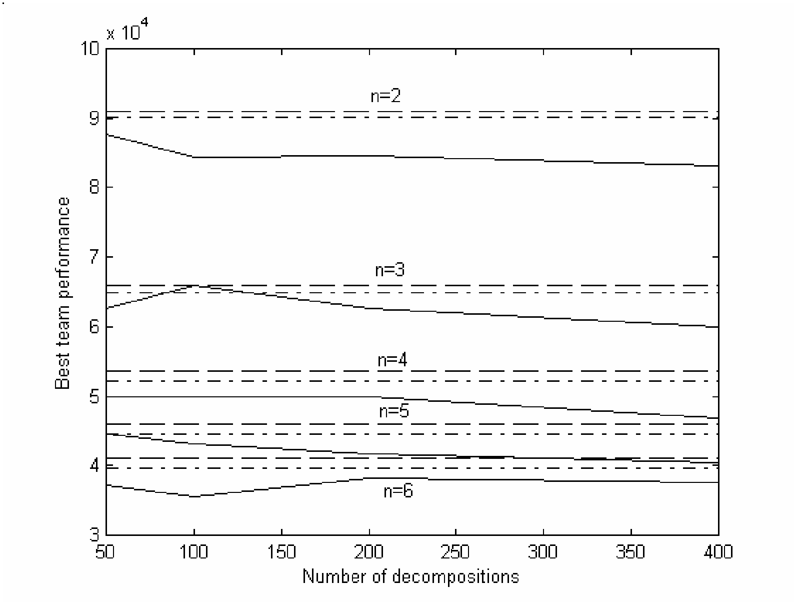

Figure 8. The best team performance for all teams and all decompositions is depicted for the case when the agents' initial locations are not correlated with the locations of their assigned cells. The dashed-lines (dash doted-lines) represent the team performance for equal-sized horizontal (vertical) cell distribution

	#	Max.	Min.	Avg.	Std. Dev.
Increases	37	17	1	7.1	4.6
Decreases	12	-13	-1	-3.4	3.3

Table 6. Experimental results for dynamic decomposition. The maximum, minimum, average and standard deviation of the increase or decrease in the percentage of the area surveyed after dynamic decomposition is shown. The total number of decompositions to which the method is applied is 49 (37+12)

with the new decomposition. If some agent is still short of power, this process is repeated. The site locations of the agents with surplus power can be moved until the agents with surplus power run short of power themselves. The process is stopped when one of the following two conditions occurs:

1. All the agents can perform their assigned sub-task.
2. The site location of none of the agents with surplus power can be moved any closer due to impending power shortage.

Experiments are performed to evaluate the proposed method. A team of agents is assigned to survey an area. Initial decomposition is performed by static decomposition but one of the agents has only enough power to perform 50% of the sub-task. The decomposition is then dynamically modified as explained. To assess the performance of the proposed method, the percentage of the overall area surveyed by the team before and after applying the dynamic decomposition is calculated and compared. The experiment is performed for 50 different initially static VD decompositions. In 37 cases the percentage of the area surveyed is increased and in 13 cases that percentage is decreased. The decrease in 8 of the 13 cases is negligible (less than 5%).

One pathological case resulting in a major decrease was noticed in the experimental results; this occurred when one of the Voronoi cells was much larger than the other cells. Applying dynamic decomposition results in the large cell becoming even larger. Since the initial static decomposition was a bad decomposition to begin with, this special case is not taken into account when the method is being evaluated. The decrease in the other four cases is not major. The maximum, minimum, average and standard deviation of the increase or decrease in the percentage of area surveyed is shown in Table 6.

7 Conclusions and Future Work

This paper proposes a method for task decomposition among a team of agents by using the VD. There are two phases to the proposed method: static and dynamic decomposition. Simulation tools were developed to run experiments and an algorithm was proposed for VD computation, which provides additional geometric information necessary for the task allocation and is also more intuitive than some existing algorithms. The first set of experiments shows that the result of static decomposition compares favorably with the base-line decomposition that leads to the team finishing in the shortest amount of time. The second set of experiments demonstrates that when some of the restrictions imposed in the first set of experiments are relaxed, static decomposition performs significantly better than the intuitive solutions. The results obtained in these experiments demonstrate that the proposed method is principled to the extent that it can improve over other reasonable and intuitive approaches with respect to the constrained resource. The third set of experiments shows that dynamic decomposition can improve the team performance when some agents in the team are experiencing power shortage. The movement of the site location of agents with surplus power is made more principled by considering their distance to the site location of agents with power shortage and also by considering the amount of their surplus power. This will localize the effect of dynamic decomposition, with the neighboring agents carrying most of the burden, and will result in faster convergence of the algorithm.

This paper shows that VD can be applied to task decomposition among a team of agents and that the methods work well empirically and can achieve good results in conserving the resource considered critical with minimal computational overhead. The focus of the paper is not on finding the optimal decomposition or on finding it in the most computationally efficient way. The focus is a proof-of-concept – VD can be applied to task decomposition problems. For the class of problems discussed in this paper, computational burden is not the limiting resource – it is time or some other resource or a combination of resources related to the agents' primary task. If the computational resources become the limiting factor, each of the approaches in the paper can be optimized to lower the computational burden.

The following is a list of possible future research:

1. Static decomposition can be applied to n-dimensional task spaces.
2. Static decomposition can be applied to different mediums, i.e. to any abstract space that can be represented as a geometric entity. One such abstract space might be the World Wide Web. For example, the method might be used for task decomposition among a team of agents that are responsible for searching the web. When a query is submitted to a search engine, the search engine generates multiple agents that will each search a different part of the web.
3. Dynamic decomposition can be generalized based on any agent resource, not just based on agent power as discussed in this paper.
4. The idea of Evolutionary Computation can be used to improve the results of the proposed task decomposition method. Beginning with a random population of VD, the fitness of each can be determined by evaluating the team performance in simulation. Mutation (or other appropriate genetic operators) can be applied to the population of the VD to guide the search for a better partitioning of the task.

8 Acknowledgement

This material is based on work carried out at the Applied Research Laboratory at The Pennsylvania State University under the ONR grant N000149615026.

References

Allen, P., I. Stamos, A. Gueorguiev, E. Gold, and P. Blaer (2001). Avenue: Automated site modeling in urban environments. In *Proc. of 3rd International Conference on 3D Digital Imaging and Modeling*, pp. 357–364.

Baccelli, F., M. Klein, M. Lebourges, and S. Zuyev (1997). Stochastic geometry and architecture of communication networks. *Journal of Telecommunication Systems 7*, 209–227.

Bose, N. and A. Garga (1993). Neural network design using voronoi diagrams. *IEEE Transactions on Neural Networks 4*(5), 778–787.

Cohen, P. R. and H. J. Levesque (1991). Teamwork. *Nous 25*(4), 487–512.

Cohen, P. R., H. J. Levesque, and I. Smith (1997). On team formation. In J. Hintikka and R. Tuomela (Eds.), *Contemporary Action Theory*, pp. 87–114.

de Berg, M., M. van Kreveld, M. Overmars, and O. Schwarzkopf (2000). *Computational Geometry, Algorithms and Applications*. Springer-Verlag.

Garga, A. K. and N. K. Bose (1994). A neural network approach to the construction of delaunay tessellation of points in rd. *IEEE Transactions on Circuits and Systems I 41*(9), 611–613.

Grosz, B. J. (1996). Collaborative systems: Aaai-94 presidential address. *AI Magazine 2*(17), 67–85.

Grosz, B. J. and S. Kraus (1996). Collaborative plans for complex group action. *Artificial Intelligence 86*(2), 269–357.

Imielinska, C., M. Downes, and W. Yuan. (2000). Semi-automated color segmentation of anatomical tissue. *Computerized Medical Imaging and Graphics 24*, 173–180.

Levesque, H., P. R. Cohen, and J. T. H. Nunes (1990). On acting together. In *Proc. of AAAI-90*, pp. 94–99.

Luchnikov, V., N. Medvedev, and M. Gavrilova (2001). The voronoi-delaunay approach for modeling the packing of balls in a cylindrical container. In *Proc. of International Conference on Computational Science (1)*, pp. 748–752.

Okabe, A., B. Boots, K. Sugihara, and S. Chiu (2000). *Spatial Tessellations, Concepts and Applications of Voronoi Diagrams*.

Sambridge, M., J. Braun, and H. McQueen (1995). Geophysical parameterization and interpolation of irregular data using natural neighbors. *Geophysical Journal International 122*, 837–857.

Stone, P. and M. Veloso (1999). Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence 110*(2), 241–273.

Stone, P. and M. Veloso (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robotics 8*(3), 345–383.

Tambe, M. (1997). Towards flexible teamwork. *Journal of Artificial Intelligence Research 7*, 83–124.