

The Cooperative Hunters — Efficient Cooperative Search For Smart Targets Using UAV Swarms

Yaniv Altshuler¹, Vladimir Yanovsky¹, Israel A. Wagner^{1,2}, and Alfred M. Bruckstein¹

¹ Computer Science Department, Technion, Haifa 32000 Israel
{yanival, volodyan, wagner, freddy}@cs.technion.ac.il
² IBM Haifa Labs, MATAM, Haifa 31905 Israel
wagner@il.ibm.com

Abstract. This work examines the *Cooperative Hunters* problem, where a swarm of UAVs (unmanned aerial vehicles) is used for searching after one or more “smart targets” which are moving in a predefined area, while trying to avoid detection by the swarm. By arranging themselves into an efficient flight configuration, the UAVs optimize their integrated sensing capability, and are thus capable of searching much larger territories than a group of uncooperative UAVs. The problem was introduced in [1], while similar work also appears in [6–10]. This work presents two decentralized cooperative search algorithms which demonstrate major improvements over the algorithm and analysis presented in [1]. The first algorithm uses improved flying patterns which achieve superior search performance. An analytic optimality proof for the algorithm’s performance is presented. The second algorithm is a fault tolerant algorithm which allows the UAVs to search in areas whose shapes and sizes are unknown to the UAVs in advance (in comparison to the algorithm of [1] which is designed for rectangular shapes whose dimensions are known to the swarm).

Keywords — Swarm Algorithm, Cooperative Search, UAVs Swarm.

1 Introduction

Significant research effort has been invested during the last few years in design, analysis and simulation of multi-agents systems design for searching areas (either known or unknown) [6–10]. While in most works the targets are assumed to be idle, recent works consider dynamic targets, meaning — target which by detecting the searching agents from a long distance, try to avoid detection by evading the agents.

Such problem is presented in [1], where a swarm of UAVs (unmanned aerial vehicles) is used for searching after one or more evading “smart targets” (i.e. a platoon of T-72 tanks, a squad of soldiers, etc’). The UAV swarm’s goal is to find the target (or targets) in the shortest time possible. Meaning, the UAV swarm must guarantee that there exist time t in which all the escaping targets are detected, and that this t is minimal. While the swarm comprises relatively simple UAVs, which lack prior knowledge of the initial positions of the targets, the targets possess full knowledge of the whereabouts of

the swarm's UAVs. In the current work, the problem described above will be denoted as the *Cooperative Hunters* problem. The search strategy suggested in [1] defines *flying patterns* which the UAVs will follow, which are designed for scanning the (rectangular) area in such a way that the targets cannot re-enter sub-areas which were already scanned by the swarm, without being detected by some UAV.

Note that this protocol assumes that the area in which the targets can move is known to the UAVs in advance, and must be rectangular in shape. Furthermore, although trying to minimize the communication between the swarm's UAVs, the presented algorithm still requires a relatively high amount of explicit cooperation between the UAVs, which can be obtained through the use of a relatively high amount of communication.

This work presents two decentralized cooperative search algorithms which demonstrate major improvements over the algorithm and analysis presented in [1]. The first algorithm, discussed in section 2 uses improved flying patterns which enables the swarm to achieve superior search performance. The optimality of this search scheme is proved in section 2.5.

The second algorithm, which is discussed in section 3, is based on a dynamic cleaning algorithm presented in [4]. The algorithm assumes no previous knowledge considering the area to be searched, and uses only a limited communication between the UAVs. The algorithm is also highly fault-tolerant, meaning — even if many UAVs will malfunction or be shot down, the swarm will still be able to complete the task, albeit slower. Analytic bounds over the search time of this algorithm are demonstrated. In addition, experimental results for the algorithm are presented.

2 HUNT-I Algorithm — Optimality in Rectangular Shapes

2.1 Notations

Below we define some notations used in this section:

- N — number of UAVs.
- D — sensor recognition diameter.
- S — line formation's scan width ($S = N \cdot D$).
- L — length of the rectangular region.
- W — width of the rectangular region.
- V — the speed of UAV.
- v — the speed of the target.

2.2 Improving the UCLA Algorithm — Motivation

In Wincent et al. [1] algorithm the UAVs were designed to fly upwards and downwards in line formations parallel to the boundary of the region. In order to ensure that no target escapes into an already clean area, the rectangles formed by consecutive sweeps must overlap. Fig. 2.2 illustrates the algorithm. W.l.o.g., suppose $S = 1$. Since it takes $\frac{2W+a}{V}$ for the UAVs to go from one side of the rectangle to the opposite side, shift rightwards a and return, in order to ensure that the target do not move into already cleared region while the UAVs are away, the rectangular scan bands need to have overlap of size $\frac{(2W+a) \cdot v}{V}$. Since the overlap's width is at most 1, the algorithm of Wincent et al. [1] cannot complete the mission in case $\frac{V}{v} \leq 2W$.

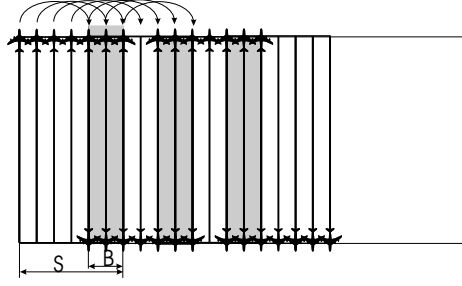


Fig. 1. Illustration of the algorithm from Wincent et al. [1].

2.3 Description of the Algorithm

Next, we show an algorithm that can guarantee locating the targets if $\frac{V}{v} \geq W + 1$. Similarly to the algorithm of Wincent et al. [1] ours uses upwards and downwards sweeping by a line formation of UAVs. Again, we assume that $S = 1$.

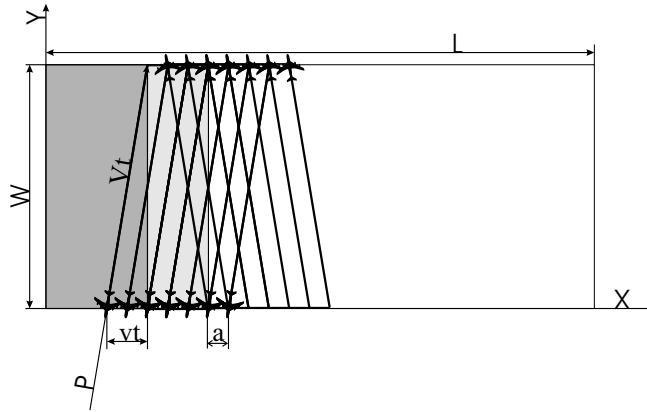


Fig. 2. Illustration of our improved algorithm.

The idea of the algorithm is illustrated in Fig. 2. Before each sweeping pass the UAVs are placed in a line at either top or bottom side of the rectangle. The area to the left of the UAVs is already clean. In Fig. 2 we show two sweeping passes. Before the first and after the second pass the planes are at the top side of the rectangle; a light-gray rectangle shows the addition to the clean area after the two passes.

Let us describe one pass of the algorithm. First, line formation moves from BC to AD . Before the stage begins at time $t = 0$, area to the left of line BF , filled with dark gray in Fig. 2, is already clean. Point A to which the left edge of UAV formation heads is chosen such that $\frac{|AB|}{V} = \frac{|AF|}{v}$. Denote $T = \frac{|AB|}{V} = \frac{|AF|}{v}$.

Lemma 1. *When the formation reaches the lower side of the rectangle at time $t = T$, the region to the left of line DJ is clean.*

Proof. We separate the proof into two parts:

1. – No target can move to the left of line AB . Indeed, suppose a target willing to cross the line is at point Q , see Fig. 3. By the choice of A , if at time $t = 0$ the target was in $Q \in [BF]$ and the UAV at B , they reach location Z , where Z is such that ZQ is parallel to axis X , simultaneously and the target will not cross AB undetected.
 - If the target attempts to cross line AB at Z' (or Z''), then it will not be able to do this, since even a target located in Q' (or Q'') at $t = 0$ is not able to do this despite being nearer to Z' (or Z''), respectively, than Z .
2. A target cannot move into $ABJD$. Indeed if it waits at line JD till the UAV formation passes and then heads leftwards, by the choice of point A the target will reach line JD at the same time as the formation reaches the bottom side of the area.

Now when the formation is at the segment AD , it needs to shift rightwards to occupy segment EG such that the area to the left of E is clean. Hence E is chosen such that $\frac{|AE|}{V} = \frac{|ED|}{v}$. This completes the description of one pass of the algorithm — as before the pass started, the region to the left of E is clean.

2.4 Analysis of the Algorithm

In this section we shall analyze the behavior of the algorithm for different values of V , v and W . First, let us see for which ratio $\frac{V}{v}$, given W , the algorithm is able to move forward.

Lemma 2. *If $\frac{V}{v} \geq W + 1$ the formation is able to move forward and complete the mission.*

Proof. In order for the algorithm to be able to move forward by some amount $a > 0$ the UAV formation must be able to shift from AD to EG faster than the “target contaminated front” reaches from DJ to KE , see Fig. 2. Hence, it must hold that $\frac{|AF|+a}{V} > \frac{|FD|-a}{v}$, when A is such that $\frac{|BA|}{V} = \frac{|FA|}{v}$. That is, for the maximal possible propagation A we have

$$\begin{cases} W^2 + (vT)^2 = (VT)^2 \\ \frac{vt+a}{V} = \frac{1-vt-a}{v} \end{cases} \quad (1)$$

After some tedious calculations and denoting $r = \frac{V}{v}$ we get

$$a(r, W) = \frac{r}{r+1} - \frac{W}{\sqrt{r^2-1}} \quad (2)$$

If r is such that $a(r, W) > 0$, the algorithm is able to proceed and will be able to complete the mission.

Following some more computations we obtain a simpler looking lower bound on propagation after a single pass:

$$a(r, W) > \frac{r}{r+1} \left(1 - \frac{W}{r-1}\right) \quad (3)$$

Hence, for $r \geq W + 1$ the propagation is positive.

Now returning to the original notations, given a sensor recognition diameter D , $y \geq W + 1$ is equivalent to the condition on the number of UAVs sufficient to guarantee the completion:

$$N = \frac{W}{(r-1)D} \quad (4)$$

This can be compared to the bound:

$$N = \frac{2W}{rD} \quad (5)$$

from Wincent et al. [1].

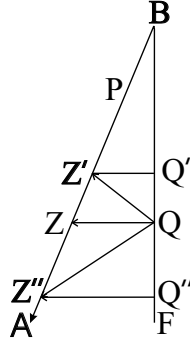


Fig. 3. Illustration of a target attempting to cross into already clean region.

2.5 Lower Bound on the Number of UAVs — Optimality Proof

After we found in the previous section that if $r \geq W + 1$, mission completion is guaranteed, we would like to prove the following lemma.

Lemma 3. *If $r < W$, the UAVs will not be able to complete the mission, independently of the algorithm they use.*

Proof. Given an algorithm, denote with $C(t)$ the convex hull of the region guaranteed to be clean of targets at time t , by $S(t)$ denote its area, and let us consider $\frac{\partial S}{\partial t}$. We show

that if t is such that $S(t) = \frac{LW}{2}$, $\frac{\partial S}{\partial t} < 0$, proving that the algorithm will not be able to complete the mission.

Denote with $P(t)$ the length of the circumference of $C(t)$ that is not part of the rectangle's boundary. From geometric considerations, it can be seen that

$$S(t + \Delta t) - S(t) < \Delta t S \cdot V - \Delta t P(S(t)) \cdot v \quad (6)$$

Recalling our assumption $S = 1$, we have

$$\frac{\partial S}{\partial t} \leq v \cdot r(r - P(S(t))) \quad (7)$$

If we prove that for t_0 s.t. $S(t_0) = \frac{LW}{2}$ it holds that $P(S(t_0)) \geq W$, the claim of the lemma will follow from the assumption $W > r$. The proof of the later is by consequently applying elementary geometric arguments to the cases of $C(t)$ having common points with 1 to 4 sides of the rectangle and is omitted here. For instance, if $C(t_0)$ has common points with one side of the rectangle, then the circumference of $C(t)$ is at most $2P(t_0) \leq 2W$. Since of all shapes with the same area a circle has the greatest area, $S(t_0) \leq \frac{W^2}{\pi} < \frac{W^2}{2}$.

3 HUNT-II Algorithm — Simplicity and Robustness

3.1 Motivation

As mentioned in previous sections, the search protocol presented in [1] assumes that the swarms' UAVs have prior knowledge regarding the area in which the targets move and hide. This knowledge is being used in order to plan the best searching strategy. However, this may not always be the case — a swarm of UAVs may be launched into an area whose shape and size is unknown to the UAVs prior to the beginning of the operation. For example, a swarm of UAVs might be used in order to hunt guerrilla forces, hiding in a city. These forces can be identified by the UAVs, since they use certain vehicles, or carrying certain electronic equipment or weapons, which can be identified by the UAVs' sensors.

In the previous example, it is known that the targets can move only within the boundaries of the city. However, the operators of the system may lack information regarding those boundaries (since for example, they may possess only old and outdated satellite photos of the area). Since the UAVs are discovering the boundaries of the area only as they explore it, a search method which does not rely on previous knowledge of the searched area must be composed.

The searching protocol presented in section 3 assumes no previous prior knowledge of the area. The only assumption made is that the swarm's UAVs are capable of identifying the boundaries of the searched area, as they pass over them. In the previous example this means that once a UAV passes over the boundary of the city (the area in which the city "ends"), it can detect this boundary (probably by using optical or other sensors). Therefore, a swarm whose UAVs contains no knowledge regarding the area to be searched, can still perform the searching task, by employing this protocol.

Furthermore, the presented protocol is very efficient in means of the simplicity of the UAVs, and the low communication between them. In the spirit of [2] we consider simple robots, with limited resources. Since UAVs contain large memory and computational resources, this assumption may seem at first out of place. However, such a UAV must perform numerous tasks simultaneously, some of which (such as visual processing) may require large memory and computational resources. This makes the use of a simple protocol a notable advantage.

The communication between the UAVs which is required for the algorithm is extremely limited, and is bounded by $6 \cdot k$ bits per time step (k being the number of UAVs). Another advantage of the presented protocol is its fault tolerance. Even should some UAVs malfunction, the swarms will still be able to complete the mission if possible, albeit with a decrease in performance.

3.2 Problem Formulation

Let us assume the time is discrete, while every time step lasts c_{time} seconds. Let G be a two dimensional grid, whose every vertex corresponds to an area in the map of size $c_{size} \times c_{size}$ square meters. There is no limitation over the shape of the area to be searched, although it is assumed to be simply connected. Let us assume that each UAV can move at $1 \cdot c_{size}$ meters per time step. Let us assume that the targets move at speed $v_{target} \cdot c_{size}$ meters per time step (thus, c_{time} can be adjusted accordingly).

We assume that each UAV is equipped with sensors able of detecting the targets within its current vertex of G . The targets however, can spot the searcher from a distance (far beyond the searcher's sensors' range) and subsequently, manoeuvre in order to evade detection. Once the searcher detects the target, it intercepts it automatically.

Each UAV is aware of its current location (using a GPS receiver) and while flying over vertex v , can identify whether v and its neighbors are part of the area to be searched, or not. The UAV's communicate between one another using a wireless channel, while the information transmitted over this channel should be kept to a minimum. The number of hiding targets can be either known to the swarms (meaning that the UAVs will stop searching once the targets have been detected), or alternatively, the UAVs might not know the exact number of hiding targets (in which case they will continue searching until guaranteeing that there is no longer a place in which the targets can hide). The goal of the swarm is to detect all hiding targets, within in a minimal amount of time.

3.3 General Principle

Although the initial problem is that of searching for hiding targets within a given area, we shall consider an alternative, yet equivalent problem — the *dynamic cooperative cleaners* problem. The static variant of the cooperative cleaners problem is described and analyzed in [3] (when a similar work may be found in [11]), while the dynamic variant of the problem is described in [4].

This problem assumes a grid, part of which is 'dirty', where the 'dirty' part is a connected region of the grid. On this dirty grid region several agents move, each having the ability to 'clean' the place (vertex, 'tile', 'pixel' or 'square') it is located in. The

dynamic variant of the problem involves a deterministic evolution of the environment, simulating a spreading *contamination* (or spreading *fire*).

Notice that from a cleaning protocol which is used by agents in order to solve the cooperative cleaners problem, a protocol for the swarm search problem can be derived. This is done by defining the entire area G as ‘contaminated’. A ‘clean’ square (either a square which has not been contaminated yet, or a square which was cleaned by the UAVs) will represent an area which is guaranteed not to contain any target. By using the fact that the contamination is spreading, we simulate the fact that the targets may manoeuvre around the UAVs, in order to avoid detection — if vertex v is contaminated then it may contain a target, thus, after $\frac{1}{v_{target}}$ seconds, this target could have moved from v to its neighbors, had it been in v . As result, after $\frac{1}{v_{target}}$ seconds all the neighbors of v become contaminated. In other words, the spreading contaminated simulates a *danger diffusion* that represents the capability of a square to contain a target.

The agents’ goal is to eliminate the contaminated area — eliminate the places which the targets may be hiding in. Once there are no longer squares in which the targets may be hiding, the swarm is guaranteed to have detected all evading targets. Note that our demands regarding the UAVs having no prior knowledge of the area to be searched are met, since the cooperative cleaners problem do not assume such knowledge.

3.4 Swarm Search Algorithm

Let each UAV i hold G_i — a bitmap of G . Let every G_i be initialized to zeros (e.g. “clean”). Let each UAV i contain a hash table of vertices — f_i which for every vertex can return *on* or *off*. The default for all the vertices is *off*. Notice that the size of the area the UAVs can move at is not limited. Once new vertices are met, they are being added to the list, and the size of G_i is dynamically increased. The list f_i represents the vertices which are known to be within the area to be searched.

Every time a UAV flying over vertex v identifies v or one of its neighbors to be a part of the area to be searched, if $f_i(v) = off$ it sets the corresponding vertices of G_i to 1, sets $f_i(v)$ to be *on*, and broadcasts this information to the other UAVs. Once a UAV receives a transmission that vertex v is part of the area to be searched, it sets $f_i(v)$ to *on* and sets the corresponding vertex in G_i to 1.

Every time a UAV moves it broadcasts the direction of its movement to the rest of the UAVs (north, south, west or east, also referred to as *up*, *down*, *left* and *right*).

Notice that every time step each UAV broadcasts the new squares which are parts of G (which are set to 1 in G_i), and the squares it “cleaned” by flying over them (which are set to 0). Thus, the G_i and f_i of all UAVs are kept synchronized. Since v_{target} is known to the UAVs, they can simulate the spreading contamination, by performing $(\forall v \in G_i, \forall u \in Neighbors(v) : state(u) = 1)$ every $\frac{1}{v_{target}}$ time steps. Thus, the bitmaps G_i always represent the correct representation of the area still to be cleaned.

The direction of movement and the decision whether or not to clean a vertex are determined using some cleaning protocol.

Whenever a UAV cleans a certain vertex, it sets this vertex in G_i to be 0, and broadcasts this information. Once a UAV receives such a transmission, it sets the vertex corresponding to the new location of the transmitting UAV to 0.

The UAVs are assumed to be placed on the boundary of the area to be searched. Thus, each G_i immediately contains at least one vertex whose value is 1. As a result, for G_i to contain only zeros, the UAVs must have visited all the vertices of G and made sure that no target could have escaped and “re-contaminated” a clean square. When G_i becomes all zeros UAV i knows that the targets have been found, and it can stop searching.

Since each time step, each UAV can move in at most 4 directions (i.e. 2 bits of information), clean at most a single vertex (i.e. 1 bit of information), and broadcast the status of 8 neighbor vertices (i.e. 3 bits of information), the communication is limited to 6 bits of information per UAV per time step.

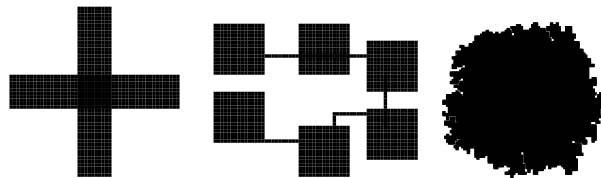
3.5 Cleaning Protocol

For solving the *Dynamic Cooperative Cleaners* problem the **SWEEP** protocol suggested in [4] is employed. The protocol can be described as follows. Generalizing an idea from computer graphics (which is presented in [5]), the connectivity of the *contaminated* region is preserved by preventing the agents from cleaning what is called *critical points* — points which disconnect the graph of contaminated grid points. This ensures that the agents stop only upon completing their mission. An important advantage of this approach, in addition to the simplicity of the agents, is fault-tolerance — even if almost all the agents cease to work before completion, the remaining ones will eventually complete the mission, if possible. At each time step, each agent cleans its current location (assuming this is not a critical point), and moves to its *rightmost* neighbor — a local movement rule, creating the effect of a clockwise traversal of the contaminated shape. As a result, the agents “peel” layers from the shape, while preserving its connectivity, until the shape is cleaned entirely.

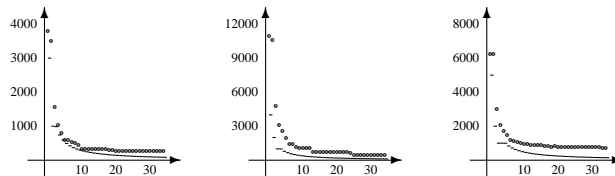
3.6 Results

In the previous sections we have shown that the swarm search problem can be reduced to the dynamic cooperative cleaners problem. Using this reduction, we are able to use protocols which were designed for the cleaning problem, and be assured that once the cleaning problem is completed successfully, so is the corresponding hunting problem. Several analytic bounds for the cleaning time of agents employing the **SWEEP** cleaning protocol in a given shape are presented in [4].

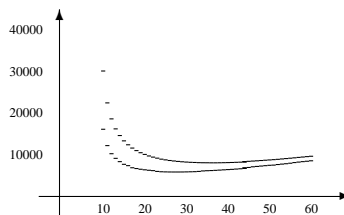
Since the performance of UAVs utilizing the **SWEEP** protocol for the hunting problem are identical to those of cleaning agents using the protocol for the dynamic cleaning problem, all the experimental and analytic results from [4] and other works concerning the dynamic cleaning problem and the **SWEEP** protocol are also applicable in our search algorithm. The simulations of [4] examined shapes of various geometric features, and tracked the search time of k agents ($k \in [1, 100]$) which used the protocol :



The lower curves represent the results predicted by the lower bound, while the upper curves represent the actual search time, produced by the simulations performed (the graphs present the search time as a function of the number of agents). The left graph presents the results that were produced by the “cross”, etc’.



The following graph contains the upper bound for the “cross”. Notice that the lower, tighter, curve was produced when taking into account that the “cross” shape is “convex”:



References

1. Vincent, P., Rubin, I.: “A Framework and Analysis for Cooperative Search Using UAV Swarms”, ACM Symposium on applied computing, (2004).
2. Breitenberg, V.: “Vehicles”, MIT Press (1984).
3. Wagner, I.A., Bruckstein, A.M.: “Cooperative Cleaners: A Case of Distributed Ant-Robotics”, in “Communications, Computation, Control, and Signal Processing: A Tribute to Thomas Kailath”, Kluwer Academic Publishers, The Netherlands, pp. 289–308, (1997).
4. Altshuler, Y., Bruckstein, A.M., Wagner, I.A.: “Swarm Robotics for a Dynamic Cleaning Problem”, in “IEEE Swarm Intelligence Symposium”, (2005).
5. Henrich, D.: “Space Efficient Region Filling in Raster Graphics”, The Visual Computer, pp. 10:205–215, Springer-Verlag, (1994).
6. Passino, K., Polycarpou, M., Jacques, D., Pachter, M., Liu, Y., Yang, Y., Flint, M. and Baum, M.: “Cooperative Control for Autonomous Air Vehicles”, In Cooperative Control and Optimization, R. Murphey and P. Pardalos, editors. Kluwer Academic Publishers, Boston, (2002).
7. Polycarpou, M., Yang, Y. and Passino, K.: “A Cooperative Search Framework for Distributed Agents”, In Proceedings of the 2001 IEEE International Symposium on Intelligent Control (Mexico City, Mexico, September 5–7). IEEE, New Jersey, 1–6, (2001).
8. Stone, L.D.: “Theory of Optimal Search”, Academic Press, New York, (1975).
9. Koopman, B.O.: “The Theory of Search II, Target Detection”, Operations Research 4, 5, 503–531, October, (1956).
10. Koenig, S., Liu, Y.: “Terrain Coverage with Ant Robots: A Simulation Study”, AGENTS’01, May 28–June 1, Montreal, Quebec, Canada, (2001).
11. Rekleitisy, I., Lee-Shuey, V., Peng Newz, A., Chosety, H.: “Limited Communication, Multi-Robot Team Based Coverage”, Proceedings of the 2004 IEEE International Conference on Robotics and Automation, New Orleans, LA, April, (2004).