Optimal Multi-Sensor based Multi Target Detection by Moving Sensors to the Maximal Clique in a Covering Graph

Ganesh P Kumar and K Madhava Krishna
International Institute of Information Technology
Robotics Research Center
Gachibowli, Hyderabad, India 500 032.
ganesh@students.iiit.ac.in, mkrishna@iiit.ac.in

Abstract

Different methodologies have been employed to solve the multi-sensor multi-target detection problem in a variety of scenarios. In this paper, we devise a time-step optimal algorithm for this problem when all but a few parameters of the sensor/target system are unknown. Using the concept of covering graph, we find an optimum solution for a single sensor, which is extended to multiple sensors by a tagging operation. Both covering graph and tagging are novel concepts, developed in the context of the detection problem for the first time, and bring a mathematical elegance to its solution. Furthermore, an implementation of the resulting algorithm is found to perform better than other notable approaches. The strong theoretical foundation, combined with the practical efficacy of the algorithm, makes it a very attractive solution to the problem.

1. Introduction

The problem of multi-sensor target tracking has attracted a significant amount of interest over the past few years. The papers [Jung and Sukhatme, 2002] and Parker[2002] are especially relevant in the context of this paper as representative algorithms; we appraise these in specific scenarios in a subsequent section. Other pertinent approaches include a scheme for delegating and withdrawing robots to and from targets through the ALLIANCE architecture [Parker 1999], a strategy for maximizing coverage with mobile sensors [Sameera and Sukhatme 2004] and a method for sensor management in a distributed tracking setup [Horling et. al, 2003]. There is also a large volume of work in the context of using a network of static sensors to guide robots pursuing invaders [Schenato et. al 2005]. A more detailed citation of literature is avoided here due to strict page limits.

Our first objective in this paper is to analyze performance of existing solutions, and reformulate target detection as an optimization problem. Subsequently, we introduce the notion of covering graph to optimize the detections of a single sensor. Extending this technique to multiple sensors is in general nontrivial, involving searching over a discrete search space, which is accomplished by a tagging operation. These ideas are crucial in transforming a continuous search space into a discrete one, enabling us to find an optimal solution from among several disparate ones. In contrast almost all existing solutions employ some heuristic to determine a good solution in an uncountable search space. Finally, we test our algorithm against two others whose parameters are tweaked by hand to perform optimally. Nonetheless, our results show that our algorithm is able to perform better than compared approaches across a number of test cases.

2. Analysis of Existing Algorithms

In this section we examine the performance of two existing representative algorithms, those of [Parker 2002] and [Sukhatme 2002].

2.1 Parker's Algorithm

Fig.1 shows a set of targets, depicted by small unshaded squares, moving counter clockwise about a circle. The shaded squares represent sensors; one sensor, closest to the circle, is repelled away by the sensor-target repulsive force referred to in [Parker 2002]. Also, every target is outside the sensing range of every other sensor. It is evident that optimal detection will occur if instead of getting repelled away, the former sensor is attracted to the inside of the circle.

2.2 Sukhatme-Jung Algorithm

Fig 2 shows an environment partitioned into five regions by landmarks represented by the diagonal black lines. A sensor S is initially stationed in region 1. Targets are located along each edge of the dotted square $ABCD$. They move in the direction indicated by the arrows cyclically with uniform velocity. The dotted circle is the FOV (field of vision) region of a nonexistent sensor placed at its center O; it is meant to touch the sides of the square, but offset for clarity. It is clear that optimal detection takes place at O. Due to the uniform velocity of sensor motion the ur-

gency of all regions will be the same, causing the sensor to either remain in region 1 or switch continually from one region to an adjacent one. Hence the algorithm is unable to find the optimal position O.

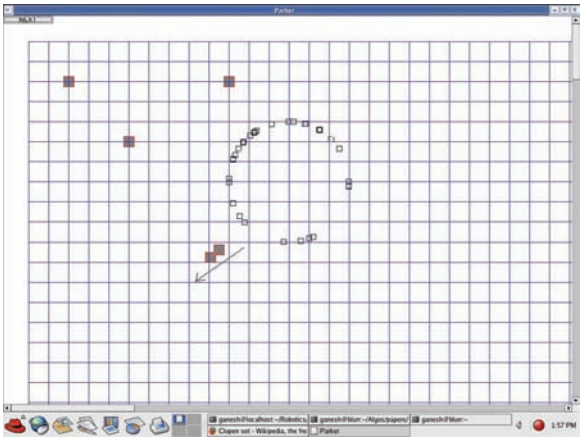


Fig. 1 A pathological run of Parker's algorithm. The shaded squares are the sensors, the unshaded ones are the targets moving in a circle. Note the repulsion of the sensor closest to the circle

Both these algorithms perform very well in many scenarios, (such as the ones cited in their sources); the pathological runs shown above are exceptions to the rule. Parker's algorithm fails when target-sensor and sensor-sensor forces give conflicting results, and Sukhatme-Jung's becomes suboptimal due to the artificial division of the environment area into regions. The presence of these exceptions leads us to ask the question: Can the performance be improved further?

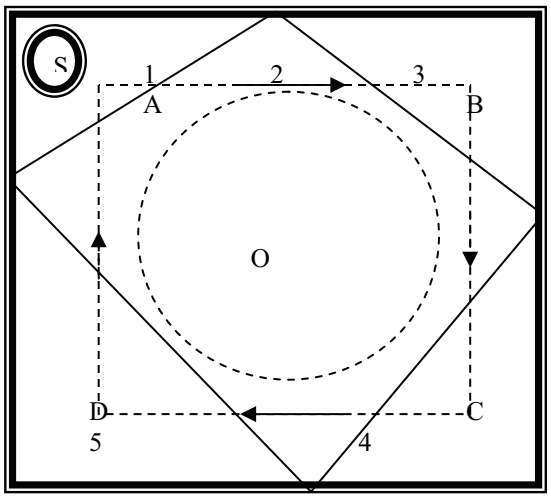


Fig. 2 Pathological run of Sukhatme-Jung algorithm. The black lines are landmarks, the dark circle S is the sensor, and the dotted lines indicate target motion paths.

3. Problem Statement

We introduce the following definitions:

1. E is a bounded region in the plane, called the *environment* over which sensors and targets move. The precise shape of E is of little significance in what follows, hence will be assumed to be an $A \times A$ square.
2. $S(t) = \{S_1(t), S_2(t).., S_n(t)\}$ is the snapshot of the set of sensors, at time instant . Each sensor has FOV $f << A$.
3. $T(t) = \{T_1(t), T_2(t).., T_m(t)\}$ is likewise the snapshot of the set of targets at time instant t.

Given $S(0)$ as a set of points within E, and $T(0)$ as a set of points on the edges of E. At every time instant $t = 1,2,3..$, every target moves with a maximum speed of $v_{\max}$ in an unknown direction. A sensor has a maximal speed of $s_{\max}$. Determine positions of $S(t)$ over time in such a way that the number of targets detected by *at least* one sensor is maximized. If there are several possible optimal solutions, we prefer those which maximize the number of detections by precisely one sensor. This last requirement minimizes redundant detections.

3.1 Preliminaries

If P is a point on the plane and $r > 0$, define $C_r(P)$ (resp. $D_r(P)$) as the circle (resp. closed disk) of radius r centered at P; also, define $\bar{D}_r(P)$ as the exterior of $D_r(P)$. Define $C(S_i)$ as the circle of radius $s_{\max}$ centered at S_i (called the *sensor circle* of S_i) and $C(T_j)$ a circle of radius $v_{\max}$ centered at T_j (called the *target circle* of T_j). Define $D(S_i)$ and $D(T_j)$ correspondingly as sensor and target disks. Note that $D_f(S_i)$ is the FOV disk of S_i. Given a set P of points on the plane, define $Hull(P)$ as the vertex set of the convex hull of P.

We call the ordered pair $(S(t_0), T(t_0))$ the *configuration* of the sensor-target system at time instant t_0, and any set of consecutive configurations a *configuration sequence*. Define a configuration sequence to be *feasible* if the following condition holds: For every pair of consecutive time instants $t', t'+1$ in the sequence, $|S_i(t'+1) - S_i(t')| \le s_{\max}$ for every sensor and $|T_j(t'+1) - T_j(t')| \le v_{\max}$ for every target; we will be interested only in feasible sequences hereafter. We define the number of *k detections* (resp. *k + detections*) of the configuration as the number of targets detected by precisely (resp. at least) k sensors, and call k the *order* of detection. With these definitions, the problem seeks to maximize 1+ detections, preferring maximal 1-detections.

3.2 Solving a set of Quadratic Constraints

This section develops a technique to be used later in Section 4.2 to assign sensor positions in a multi-sensor system. Given two disjoint sets of targets, $T_{inc} = \{T_1, T_2, .., T_k\}$ and $T_{exc} = \{T_{k+1}, T_{k+2}, ..T_l\}$, we are asked to determine whether we can place a sensor that covers all targets in T_{inc} and none of those in T_{exc}. The problem reduces to determining if there exists a sensor position P for which

$$\| P - T_i \| \leq f, \forall T_i \in T_{inc} \quad [3.2a]$$

$$\| P - T_i \| > f, \forall T_i \in T_{exc} \quad [3.2b]$$

has a solution . [3.2] has a solution if and only if the intersection $I = \bigcap_{T_i \in T_{inc}} D_f(T_i) \cap \bigcap_{T_i \in T_{exc}} \bar{D}_f(T_i)$ is non-null; in this case, every point in I is a solution. Further, suppose that [3.2a] has a solution, but with the additional constraint that $\| P - T_{k+1} \| > f$, it does not. This means that every circle of radius f which covers T_{inc} also covers T_{k+1}. In such a case, T_{k+1} is said to be *dependent* on T_{inc}; in particular, if T_{k+1} falls within $Hull(T_{inc})$, it is always dependent on T_{inc}.

4 Methodology

4.1Single Sensor System

Covering Graph

Consider the configuration $(S_1(t_0), T(t_0))$. We say that a point $P \in D(S_1)$ is a *covering point* with respect to S_1, of a set $T' \subseteq T$ of targets, currently within $D_f(S_1)$, iff no target in T can escape out of $C_f(P)$ in one time step. Observe that if P is a covering point of target $T_i \in T'$, then $P \in D_d(T_i)$, where $d = f - v_{\max}$. In the same scenario we say that every pair of targets in T' is *simultaneously coverable*. Note that two targets are simultaneously coverable iff the (Euclidean) distance between them is at most $2d$. Define the *covering graph* $CG(S(t), T(t))$ of the configuration as a graph whose nodes are covered targets, and whose edges connect a pair of simultaneously coverable targets. From above, targets in T' will form a clique in the covering graph. Further define a *maximal clique* in $CG(S(t), T(t))$ as one with the maximal number of vertices, noting that this could be non-unique.

It is easy to see that moving to a covering point of a maximal clique ensures that a maximum number of targets will be detected at the next time instant. We formalize this idea as follows: Define the *covering function* $CF : D(S_1) \rightarrow N$ which takes a point in $D(S_1)$ to the number of targets that a sensor at that point will definitely cover in the next time step. The maximal values of the covering function will be attained at the points of intersection of $C_f(T_i) \cap C_f(T_j)$, where $T_{i,j} \in D_f(S_1)$, giving us an algorithm to move S_1 to a covering center of a maximal clique. S_1 visits each of the points of intersection of the form above, and moves to that which gives maximal detections; should there be no intersections, it moves to a covering point of any target it detects. This ensures a maximal number of detections in the next time step, and the algorithm runs in $O(m^2)$ time where m is the number of targets.

Number of Cliques

In this section we estimate the maximal number of cliques that can exist in $D_f(S_1)$. A clique in $CG(S_1(t), T_1(t))$ has all its targets lying within a circle of radius $d = f - v_{\max}$ as a result, the separation between two cliques is at least d. Consequently, the number of cliques in $D_f(S_1)$ is at most the size of maximum set of points M that can be chosen in $D_f(S_1)$ such that the distance between any every point in M, with its nearest neighbors in M, is d. Such an M can be constructed as follows. Choose a point $P_0 \in D_f(S_1(t))$; draw the circle $C' = C_d(P_0)$, and mark off the vertices of a regular hexagon inscribed in it . Repeat the construction at every $P_i, 1 \leq i \leq 6$ until no more points within $D_f(S_1)$ can be found. This leads to tiling of $D_f(S_1)$ by regular hexagons of side d, leading to $O((\frac{f}{d})^2)$ hexagons and clique centers.

4.2 Multi-sensor System

In this section we use ideas from previous sections to obtain an optimal solution to tracking with n sensors. The sensors share knowledge of the targets they see at every time instant, enabling each to build the global covering graph..

Taglists and Taglist types

To optimize 1+ detections is a nontrivial problem in general. We assume for the purpose that every target T_i is tagged with a list of sensors, $T_i.taglist$. Also, define $T.taglist = \bigcup_{T_i \in T} \{(i, T_i.taglist)\}$. The point of introducing taglists- as claimed in Section 1- is to convert a continuous search space into a discrete one. When we say T_i is

tagged with a list $S' \subseteq S$, we try to see if T_i can be covered by the sensors in S'; if this is feasible, then sensor positions can be assigned to sensors in S' by the algorithm in Section 3.2, or else deemed impossible by the same. We formalize this in the following definitions.

Given $S(t_0)$, an instance of $T(t_0+1).taglist$ is said to be *feasible* iff for every sensor the intersection

$$R_i = D(S_i) \cap \bigcap_{T_j \in Cover(T.taglist, S_i)} D_f(T_j) \cap \bigcap_{T_j \notin Cover(T.taglist, S_i)} \overline{D_f(T_j)}$$

is nonempty. Here $Cover(T.taglist, S_i)$ is the set of all targets covered by S_i. We say that a point $P \in D(S_i)$ is a *covering point* with respect to S_i, of a set $T' \subseteq T$ of targets, currently detected by at least one sensor in the system, iff no target in T can escape out of $C_f(P)$ in one time step. The covering graph and covering function of a sensor are defined correspondingly.

For a single sensor, optimality of detection hinged on the assurance that a certain number of targets would be seen the next time step. This assurance is formalized by the idea of a *proper* taglist in a multi-sensor system. A feasible instance of $T(t_0+1).taglist$ is said to be *proper* iff for every sensor $S_i(t)$, targets in $Cover(T.taglist, S_i)$ are vertices of a clique in $CG(S_i(t_0), T(t_0))$. It is difficult to construct proper taglists directly ; it is easier to take a shortcut through taglists which satisfy the *hull property*, defined next.

An instance of $T(t_0+1).taglist$ is said to s*atisfy the hull property* if and only if the following criterion holds for every subset T' of targets in it. If every target in T' is tagged with sensor S_i, then every target located within $Hull(T')$ is also tagged with S_i. Note that if an instance of $T(t_0+1).taglist$ is proper, it also satisfies the hull property; the converse need not be true. We will use this property to generate proper taglists and later to compute the complexity of the multi-sensor system.

Finding the Optimal Solution

The *Findopt* routine finds the sensor assignment that guarantees maximal 1+ detections in the next time step.

$Findopt(S(t_0), T(t_0))$

1. Generate an initial proper taglist, *Current* for $T(t_0)$. Compute $N_{1+}(Current)$ and $N_1(Current)$, which denote the number of 1+ and 1 detections in *Current* respectively.
2. For each proper taglist *Next* remaining do
   a. If $N_{1+}(Next) > N_{1+}(Current)$ or ($N_{1+}(Next) = N_{1+}(Current)$ and $N_1(Next) > N_1(Current)$) then $Current \leftarrow Next$
3. Assign sensor positions corresponding to *Current* using the method in Section 3.2.

Finding all proper taglists

It remains to specify how we find proper taglists. Note that only sensors with overlapping FOV regions pose a difficulty, for otherwise we run the single-sensor algorithm on each sensor. The brute force approach would be to generate all possible taglists, check whether each is proper, and pass it to *Findopt*, at the expense of having an $O(2^{km}), k \geq 1$ algorithm. This is too expensive a solution in time. Instead, we generate all taglists satisfying the convex hull property, which in general takes time sub-exponential in m, and check whether each is proper. This is formalized below.

If $D_f(S_i) \cap D_f(S_j)) = \phi$, for every pair of distinct sensors, the only proper taglist we generate corresponds to maximal clique detections by each sensor. If the sensors move in such a way that their FOV overlaps, we degenerate to the weakly overlapping case covered below.

We call $S'(t_0) = \{S_1(t_0), S_2(t_0), .., S_k(t_0)\} \subset S(t_0)$ *strongly overlapping* iff $\bigcap_{S_i \in S'} D_f(S_i(t_0)) \neq \phi$. Suppose that $T'(t_0)$ is the set of targets detected by S', and further that $T''(t_0) \subset T'(t_0)$ is detected by more than one sensor. Suppose that $Q_i(t_0)$ is the set of cliques of $CG(S_i(t_0), T'(t_0)), 1 \leq i \leq k$.

We generate the space of proper taglist instances for $T'(t_0)$ as follows. Construct an iterator I_1 that traverses the space $Q_1(t_0) \times Q_2(t_0) \times .. \times Q_k(t_0)$. At every point of the traversal of I_1, do the following. Tag every target that appears in one or more cliques with all sensors in whose FOV it falls. Suppose that T''' is that subset of T'' which is currently detected by more than one sensor in S'. Construct iterator I_2 which goes over all possible combinations of sensor assignments to targets in $Hull(T''')$. Each of these satisfies the hull property. Output the resulting taglist if and only if it is proper.

We call $S'(t_0)$ *weakly overlapping* iff for every $S_i \in S'$ there exists at least one $S_j \in S'$: $j \neq i$ such that $D_f(S_i) \cap D_f(S_j) \neq \phi$. Following the same notation as in the previous case, we generate the space of proper taglist instances as follows. I_1 has the same meaning as before. It is then possible to write $S'(t_0)$ as $S'(t_0) = S_1'(t_0) \cup S_2'(t_0) \cup .. \cup S_r'(t_0)$, where sensors in each $S_i'(t_0)$ are strongly overlapping. Correspondingly T''' is expressible as a union $\bigcup_{i=1}^{r} T'''_i(t_0)$. Construct itera-

tor I_2 which goes over all possible sensor assignments to targets in $\prod_{i=1}^{r} Hull(T'''_i(t_0))$, the Cartesian product of hulls; output the resulting taglist iff it is proper.

Complexity Analysis

A typical worst case scenario is one in which $S(t)$ is weakly overlapping. In this case I_1 runs over $O((\frac{f}{d})^{2n})$ clique combinations, so the algorithm is exponential in the number of overlapping sensors. Suppose that there are $k : 1 \le k \le n$ hulls for I_2 to consider, and that the i-th hull encloses, or has on its boundary, m_i targets in all, and m'_i vertices. Suppose further that all m targets are currently detected. Then I_2 takes $O(n^{\sum m'_i})$ iterations to traverse its space, and each iteration takes polynomial time in m in for computing the hulls, verifying hull points, and so on. Note that $m'_i = o(m_i) = o(m)$ except in rare cases, so that the run-time is sub-exponential, typically polynomial, in the number of detected targets.

In essence while the algorithm is exponential in number of *overlapping* sensors it is typically polynomial in the number of targets due to tags for only the targets that form the convex hull for a set of targets that constitute a clique. In steady state when sensors have diverged to maximize detections, the number of overlapping sensors is generally small, and this is specifically so when the area of *E,* is considerably greater than the total area covered by the sensors. Indeed when the total area covered by sensors is greater than or equal to area of *E* maximizing detections is trivially achieved.

5. Simulation and Results

5.1 Implementation Specifics

The optimal algorithm above, hereafter called OPT, as well as the representative algorithms of Parker and Sukhatme-Jung, have been implemented on an Intel P4 machine running Fedora Core 4 using the g++ compiler and Qt for graphical simulation. The specifics of each implementation are given below.

OPT Algorithm

This is essentially the same as the multi-sensor version presented in Section 4.2 with the restriction that, when OPT runs for 1 second, it terminates and outputs the most optimal solution generated thus far. This restriction ensures that OPT takes nearly the same time to output its solution as its adversaries.

Parker Algorithm

We implemented the A-COMMT version of the Parker algorithm. We set the sensing range of every sensor to the FOV of OPT sensors during comparison; predictive tracking and communication ranges do not matter for simulation. In addition, we tweaked the values of dr_i and do_i, corresponding to cut-off points for sensor-sensor and sensor-target forces, until they gave optimal performance. Specifically, we changed do_1 and dr_1 from one test case to another, observing that $1.2do_1 < do_2 < 1.5do_1$ and $2.0do_1 < do_3 < 2.7do_1$ gave best results; likewise did $1.8dr_1 < dr_2 < 2.4dr_1$. Furthermore, we scaled the weighted force vectors to an equal degree for all sensors.

Sukhatme-Jung Algorithm

We partitioned the square environment into a cross-shaped topological map. We tweaked the urgency and availability thresholds from one test case to the next; in particular, $4 \le availability \le 10$ and $4 \le urgency \le 8$ give maximal detections. The control parameter θ_d was varied likewise across test cases till maximal detections were obtained; specifically, when the FOV or the number of sensors was decreased, we gave a small value to θ_d, between 0.5 and 0.6, and large otherwise, between 1.5 and 1.8.

5.2 Tabulated Results

The environment is the square $[0,500]\times[0,500]$. Five sensors, each of FOV 50, are randomly placed in E, each with $s_{\max} = 50$. Targets emanate from five sources placed randomly on the edges of E every time instant. The speed of each target is uniformly distributed in $[0,5]$ and angle uniformly distributed on $[0,2\pi]$. In Tables 1-3, the rows indicate the order of the detection as a percentage rounded to the nearest integer, and the columns indicate the algorithm whose performance is noted. Tables 2 and 3 show readings of the form A|B, to be interpreted as follows. In table 2, A corresponds to the case of 7 sensors, and B to that of 3 sensors, as indicated in its caption. Table 3 is read similarly.

5.3 Performance Graphs

Fig. 3a (resp. 3b) shows how the number of 1+ detections varies with FOV(resp. number of sensors) for each algorithm. The performance gain of OPT over the earlier ones is evident for various sensor and FOV combinations as it gives consistently higher 1+ detections, despite the manual tweaking of its adversaries to achieve optimal performance.

Detection	OPT	Parker	Sukhatme-Jung
0	50	60	55
1	49	37	45
2	1	3	0
3+	0	0	0

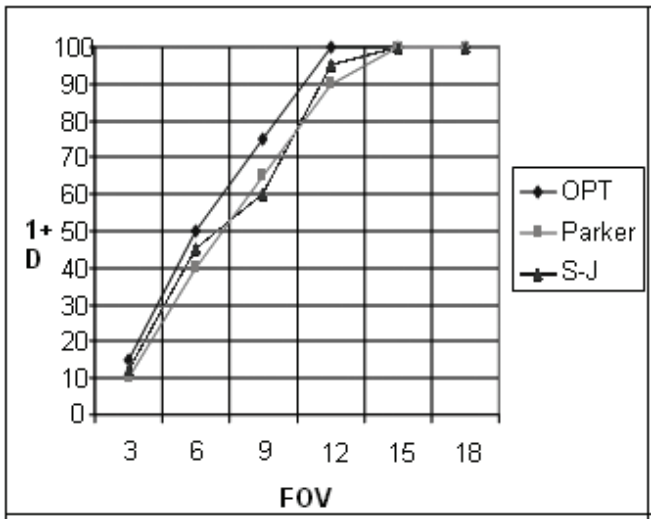
Table 1: Table of base case

Detection	OPT	Parker	Sukhatme-Jung
0	20|87	13|80	19|85
1	75|14	83|20	76|15
2	2|0	5|0	5|0
3+	1|0	0|0	0|0

Table 2: For cases: *number of sensors increased to 7 | decreased to 3*

Detection	OPT	Parker	Sukhatme-Jung
0	17|84	33|88	31|90
1	75|16	64|11	60|10
2	6|0	3|1	8|0
3+	2|0	0|0	1|0

Table 3: For cases: *Increasing FOV to 75|Decreasing to 25*



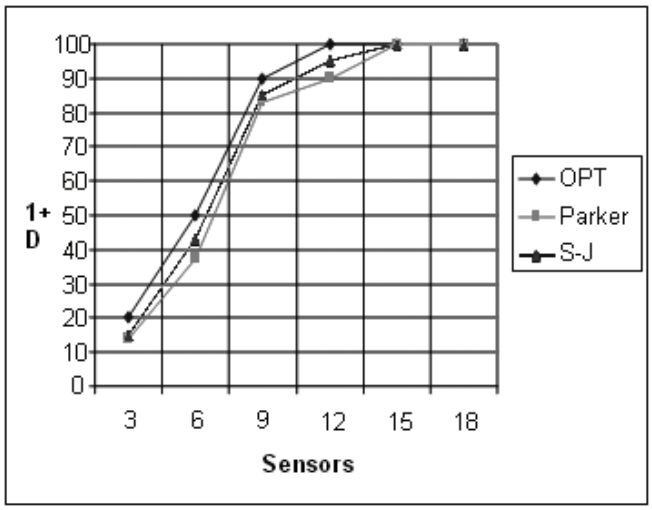


Fig. 3a (above) and 3b(below): Variation of performance with (a)FOV and (b)Number of Sensors. S-J = Sukhatme-Jung.

6. Conclusions

A novel algorithm for multi-sensor based multi target detection has been formulated and analyzed in terms of its computational complexity. Comparison with other approaches in literature verifies its efficacy across diverse scenarios. Such a multi-sensor algorithm finds applications in border patrol, guarding of secured areas, search and rescue and warehouse surveillance.

References

[Horling et. al, 2003] B Horling, R Miller, M Sims, and V Lesser, "Using and Maintaining Organization in a Large-Scale Distributed Sensor Network", In *Proceedings of the Workshop on Autonomy, Delegation, and Control, (AAMAS 2003)*.

[Jung and Sukhatme 2002] B Jung and G.S. Sukhatme, "A Region-based approach for Cooperative Multi-Target Tracking in a Structured Environment", Proc., *Proceedings of International Conference on Intelligent Robots and Systems,* 2002

[Parker1999] L Parker, "Cooperative robotics for multi-target observation", *Intelligent Automation and Soft Computing,* 5[1]:5-19, 1999

[Parker 2002] L Parker, "Distributed algorithms for multi-robot observation of multiple moving targets", *Autonomous Robots,* 12, 3 (2002)

[Poduri and Sukhatme 2004] S Poduri and G S Sukhatme, "Constrained Coverage for mobile sensor networks", *IEEE ICRA,* 165-171, 2004

[Schenato et. al, 2005] L Schenato, S Oh, S Sastry and P Bose, "Swarm Coordination for Pursuit Evasion Games using Sensor Networks", *IEEE ICRA, 2005*