On-line Parallel Heuristics, Processor Scheduling and Robot Searching under the Competitive Framework*

Alejandro López-Ortiz† Sven Schuierer‡

August 7, 2003

Abstract

In this paper we investigate parallel searches on m concurrent rays for a point target t located at some unknown distance along one of the rays. A group of p agents or robots moving at unit speed searches for t. The search succeeds when an agent reaches the point t. Given a strategy S the competitive ratio is the ratio of the time needed by the agents to find t using S and the time needed if the location of t had been known in advance. We provide a strategy with competitive ratio of $1 + 2(m/p - 1)(m/(m-p))^{m/p}$ and prove that this is optimal. This problem has applications in multiple heuristic searches in AI as well as robot motion planning. The case $p = 1$ is known in the literature as the cow path problem.

1 Introduction

Searching for a target is an important and well studied problem in robotics. In many realistic situations such as navigation in an unknown terrain or a search and rescue operation the robot does not possess complete knowledge about its environment. In the earlier case the robot may not have a map of its surroundings and in the latter the location of the target may be unknown [4, 13, 14, 19, 20].

The *competitive ratio* [22, 13] of a search strategy S is defined as the maximum of the ratio of the search cost using S and the optimal distance from the starting point to the target, over all possible positions of the target.

Consider an exhaustive search on m concurrent rays. Here, a point robot or—as in our case—a group of point robots is assumed to stand at the origin of m concurrent rays. One of the rays contains the target t whose distance to the origin is unknown. The robot can only detect t if it stands on top of it. It can be shown that an optimal strategy for one robot is to visit the rays in cyclic order, increasing the step length each time by a factor of $m/(m-1)$ if it starts with a step length of 1. The competitive ratio C_m achieved by this strategy is given by $C_m = 1 + 2m^m/(m-1)^{m-1}$ which can be shown to be optimal [2, 7, 11, 17]. The lower bound

*This research is partially supported by the DFG-Project "Diskrete Probleme", No. Ot 64/8-2.

†Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1. alopez-o@uwaterloo.ca

‡Institut für Informatik, Am Flughafen 17, Geb. 051, D-79110, Freiburg, Germany. schuiere@informatik.uni-freiburg.de

in this case has proven to be a useful tool for proving lower bounds for searching in a number of classes of simple polygons, such as star-shaped polygons [15], $\mathcal{G}$-streets [6, 16], HV-streets [5], and θ-streets [5, 9].

Parallel searching on m concurrent rays has been addressed before in two contexts. In the first context a group of p point robots searches for the target. Neither the ray containing the target nor the distance to the target are known. The robots can communicate only when they meet. The search concludes when the target is found and all robots are notified thus. Baeza-Yates and Schott investigated searching on the real line [3] and Hammar, Nilsson and Schuierer considered the same case for m concurrent rays [8].

The second context is the on-line construction of hybrid algorithms. In this setting we are given a problem Q and m heuristics or approaches to solving it. The implementation of each approach is called a *basic* algorithm. We are given a computer with $k < m$ disjoint memory areas which can be used to run one basic algorithm and to store the results of its computation. Only a single basic algorithm can be run on the computer at a given time. It is not known in advance which of the algorithms solves the problem Q—although we assume that there is at least one—or how much time it takes to compute a solution. In the worst case only one algorithm solves Q whereas the others do not even halt on Q. One way to solve Q is to construct a hybrid algorithm in the following way. A basic algorithm is run for some time, and then a computer switches to another basic algorithm for some time and so on until Q is solved. If $k < m$, then there is not enough memory to save all of the intermediate results. Hence, the current intermediate results have to be discarded and later recomputed from scratch. An alternative way to look at this problem is to assume that we are given k robots searching on m rays for a target. Each ray corresponds to a basic algorithm and a robot corresponds to a memory area, with only one robot moving at any given time. Discarding intermediate results for an algorithm A is equivalent to moving the robot on the ray corresponding to A back to the origin. Kao et al. [10, 23] gave a hybrid algorithm that achieves an optimal competitive ratio of $k + 2(m-k+1)^{m-k+1}/(m-k)^{m-k}$.

A generalization of this context is to consider a distributed setting in which more than one computer or robot perform a simultaneous search. In this case at least one of the robots must reach the target, at which time the search is considered complete. The robots move at unit speed and the competitive ratio is defined as the ratio between the search time and the shortest distance to the target. Under this framework López-Ortiz and Sweet show that the integer lattice on the plane can be searched efficiently in parallel [18].

In this paper we study searches in m concurrent rays which also correspond to an m heuristic problem with $k = p$ memory states and p processors or computers. The "terminate-on-success" framework models search and rescue operations as well as multiple heuristic searches. We provide an optimal strategy with competitive ratio of $1+2(m/p-1)(m/(m-p))^{m/p}$ for searching m rays with $p < m$ robots in parallel. The case $p = 1$ is sometimes referred in the literature as the cow path problem [11].

A variation of the strategy proposed in this paper can be applied to other graphs such as trees, resulting in an iterative deepening scheme which is optimal to within a constant factor. This shows that neither Bread First Search (BFS) nor Depth First Search (DFS) are optimal —absent any other information. This has relevance in distributed computing searches in game spaces and automated theorem proving.

In particular, computer chess gives two interesting applications for parallel searches. The first is the *three hirn* heuristic pioneered by Ingo Althöfer (see for example [1]). This heuristic uses three independent sources of advice, namely two computer programs and a person, to play a game such as chess. The goal of the three hirn team, as it is to be expected, is to find a good move for the current position. Each computer program uses its own heuristics to propose the best possible move in the time alloted. Then the person acts as an arbiter to determine which of the two programs seems to have gotten closer to the target of finding a good move and choses the best of the two [1]. A natural generalization is to use m heuristics on p processors, for $p < m$, with the processor scheduling as indicated by the strategies proposed in the present work. This results in the optimal use of computing resources, absent other information.

A second application is in the use of massively parallel chess computers, such as IBM's Deeper Blue. This computer, which defeated World Champion G. Kasparov in 1997, was equipped with 512 dedicated chess processors [12]. At a first glance one might posit that this gives a factor of 512 speed up in computations over a single processor computer. However if the exploration of the game search space tree involves iterative deepening –as it often does– our results imply that the speed advantage is in reality a multiplicative factor on the order of 2781, which is over five times higher than expected. Indeed, a parallel m ray solution would have a speed-up advantage of approximately $2\,e \approx 5.436$ above and beyond that expected from the increase in processor count alone.

The paper is organized as follows. In the next section we present some definitions and preliminary results. In Section 3 we characterize the nature of a subfamily of optimal strategies and present a lower bound for the problem of searching on m rays with p robots. In Section 4 we then present an algorithm that achieves this bound, which is optimal.

2 Preliminaries

In the following we consider the problem of a group of p robots searching for a target of unknown location on m rays in parallel. The competitive ratio is defined as the quotient of the search time over the shortest distance to the target. In this case we consider robots that have the same maximal speed, which is assumed to be, without loss of generality, one unit of distance per unit of time.

Given a strategy S, at a time T the *snapshot* of S is given by $m+2p$ values $(s_1, \ldots, s_m, d_1, I_1, d_2, I_2, \ldots, d_p, I_p)$ where s_i is the distance up to which ray i is explored, d_i is the distance of robot i to the origin, and I_i is the index of the ray that robot i is located on. Consider now a strategy S to search on m rays with p robots, in which the robots repeatedly travel one ray for a certain distance and then return to the origin to choose another ray. Let $X_S = (x_0, x_1, \ldots)$ be the collection of distances at which the robots change direction to return to the origin, ordered by the time at which the robots turn around.

Let r_i be the ray on which the robot that turns at x_i is located and T_i be the first time that a robot passes x_i on ray r_i again. Assume that this robot turns around again after having traveled a distance of x_k, where $k > i$. If the target is placed on r_i between x_i and x_k, say at

distance d after x_i, then the competitive ratio of the strategy for this placement is

$$\frac{T_i + d}{x_i + d}.$$

Since the competitive ratio is a worst case measure, we see that the competitive ratio C_S of S is at least

$$C_S \geq \sup_{d>0} \left\{ \frac{T_i + d}{x_i + d} \right\} = \frac{T_i}{x_i}. \tag{1}$$

As the target is necessarily found at some point along a step, we obtain

$$C_S = \sup_{i \geq 0} \left\{ \frac{T_i}{x_i} \right\}.$$

We say a ray r is *occupied* at time T if there is a robot on r at this time. We say a ray r is *busy* at time T if there is a robot on r that is moving away from the origin at this time. Let the *schedule* of robot R be the sequence of rays in the order in which they are explored by R together with the distance to which they are explored, i.e. $Sch_R = (d_1, I_1, d_2, I_2, \ldots)$. Given two strategies we say that S_1 is *contained* in S_2 up to time T, denoted $S_1 \subseteq_T S_2$, if the snapshots of both strategies coincide for all $t \leq T$. Given a sequence of strategies $\mathcal{V} = (S_1, S_2, \ldots)$, we say that the sequence $\mathcal{V}$ converges to a limit strategy S if there is a strictly increasing function $T(n)$ with $\lim_{n\to\infty} T(n) = \infty$ such that for each n, $S_m \subseteq_{T(n)} S_{m+1}$ for all $m \geq n$. The limit strategy S is defined in the obvious way.

3 A Lower Bound

We are interested in proving a lower bound on C_S for any on-line strategy S.

Lemma 1 *Let S be a strategy to search on m rays with p robots. Then there exist a strategy S' with the same competitive ratio or better such that*

1. *At any time t, there is at most one robot on a given ray.*
2. *If a robot moves towards the origin on some ray, then it continues until it has reached the origin.*
3. *All robots are moving at all times.*

Proof. Assume that there are at least two robots on a given ray. Either the paths of these two robots cross in opposing directions along the ray or not. In the latter case, this means that one robot trails the other along that ray and, hence, has no net effect in the exploration. Clearly a modified strategy S' in which the trailing robot stays put in the origin has the same competitive ratio as S. Alternatively, if the robot paths cross in opposing directions consider a strategy S'' which replaces the cross-paths with a "bounce", in which both robots change direction at the point of intersection of their paths. The robots also exchange schedules from that point onwards. S'' is now a strategy in which the robots never properly cross in opposing directions,

and hence itself can be replaced with a strategy S' in which one of the robots stays in the origin. S' is a strategy with the same competitive ratio as S in which robots do not change direction away from the origin.

Similarly, if the robot is moving towards the origin and then changes direction, we can create a strategy S' in which the robot stays put rather than moving toward the origin and then backtracking its steps. The strategy S' has the same competitive ratio as S, but no changes in direction away from the origin along a ray.

Lastly if we consider a robot whose sequence of moves includes a stand-still period, clearly removing those idle periods can only decrease the competitive ratio. Let R be a robot that is idle at step i. Then R moves ahead to explore ray I_i in its schedule Sch_R. However this ray might presently be occupied in which case R exchanges schedule with the robot R' occupying the ray and moves ahead to the next ray in $Sch_{R'}$. In turn, this ray might also be occupied, and the robot exchanges schedules yet again, and so on. Note that a swap on a given ray monotonically increases the distance to be traversed on that ray by it's occupant. Hence this defines a sequence of strategies whose limit strategy S' is well defined. Moreover, S' satisfies all three properties required in the lemma and has competitive ratio no larger than the original strategy S. □

Lemma 2 *There is an optimal on-line strategy to search on m rays with p robots that satisfies Lemma 1 such that if a robot is located at the origin at time T, then it chooses to explore the ray that has been explored the least among all non-busy rays.*

Proof. Let S be an optimal strategy to search on m rays with p robots that satisfies Lemma 1. Assume that robot R is located at the origin at time T and chooses to explore ray r which is explored up to distance d_r. Assume that there is a non-busy ray r' that is explored up to distance $d_{r'} < d_r$. Now consider the strategy S' where the robot chooses to explore the ray r' and the robot that explores ray r' after T in S explores ray r in S'. Each of these rays is explored in S' to its originally scheduled distance in S, only the order changes. Everything else remains the same.

The only difference in competitive ratio between the strategies S and S' is the time when the point located at a distance d_r on ray r is passed the first time by a robot and the time when $d_{r'}$ is passed the first time by a robot on ray r'.

Assume that in Strategy S a robot passes $d_{r'}$ on ray r' at time $T' + d_{r'}$. Since r is explored before ray r', we have $T' > T$. Hence, the competitive ratio of S for those two steps is

$$\max\left\{\frac{T+d_r}{d_r}, \frac{T'+d_{r'}}{d_{r'}}\right\} = 1 + \max\left\{\frac{T}{d_r}, \frac{T'}{d_{r'}}\right\}$$

whereas the competitive ratio of S' for those two steps is

$$\max\left\{\frac{T+d_{r'}}{d_{r'}}, \frac{T'+d_r}{d_r}\right\} = 1 + \max\left\{\frac{T}{d_{r'}}, \frac{T'}{d_r}\right\}.$$

Since $T' > T$ and $d_r > d_{r'}$, $T'/d_{r'} > \max\{T/d_{r'}, T'/d_r\}$ and the competitive ratio of S' is no greater than the competitive ratio of S.

This shows that switching the searching order to favour the least explored ray has no negative effect on the competitive ratio. However if the non-busy ray r' was occupied, then S' violates condition (1) of Lemma 1. In this case, rather than R exploring the occupied ray r' it exchanges schedule from that point onwards with the occupant of r' as in the proof of Lemma 1. First we observe that after the exchange of schedules, r' is no longer the least explored non-busy ray as it either has been explored to a distance $d > d_r > d'_r$ which is further than ray r or it is in the process of being explored to that distance and hence is busy. In this case, we have a new strategy S' in which robot R is about to explore a ray r' which might or might not be non-busy and occupied. We apply the same procedure to what would be the least explored ray r'' in the new strategy S' and we obtain a new strategy S'' in which ray r'' is about to be explored. Note that the distance to which r' is explored increased. Hence this creates a sequence of strategies $(S, S', S'', \ldots)$ whose limit strategy has competitive ratio no larger than S. Moreover this new strategy satisfies the properties of Lemma 1 and robots explore the least explored non-busy ray in sequence. □

Corollary 1 *There is an optimal strategy to search on m rays with p robots such that at any time the explored distances of all occupied, but not busy rays are larger than the minimum of the explored distances of all unoccupied rays.*

Proof. By Lemma 2 there is an optimal strategy such that a robot at the origin always chooses to explore the non-busy ray that is explored the least. If this ray is occupied, then there is a time at which two robots are on the same ray—a contradiction to Lemma 1. □

A strategy satisfying Lemmas 1 and 2 is termed a *normalized strategy*. The next lemma provides a lower bound for normalized optimal strategies.

Lemma 3 *The competitive ratio C_S of an optimal normalized strategy S with turn point sequence $X = (x_0, x_1, \ldots)$ is at least*

$$C_S \geq \sup_{k \geq 0} \left\{ 1 + 2 \sum_{i=0}^{k+m-p} x_i^s \Big/ \sum_{i=k-p+1}^{k} x_i^s \right\} \tag{2}$$

where $X^s = (x_0^s, x_1^s, \ldots)$ is the sequence of the sorted values of X and $x_i^s := 0$ if $i < 0$.

Proof. Let S be an optimal normalized strategy. Consider a time T such that robot R_0 is located at the origin. Since S is a normalized strategy, R_0 will explore the ray r_0 that has been explored the least among all occupied rays at time T. In general, let r_j be the current ray of robot R_j at time T, for $0 \leq j \leq p-1$.

Now consider the sequence of turn points taken by a robot R_j up to—but not including—time T. These turn points are elements in the sequence X^s; let I_j be the set of indices in X^s of these turn points of robot R_j.

Let the distance up to which ray r_0 is explored at time T be d_0. Note that $d_0 = x^s_{k_0}$, for some $k_0 \geq 0$. Furthermore, let d_j be the distance up to which ray r_j was explored before the robot R_j entered ray r_j. Note that $d_j = x^s_{k_j}$, for some $0 \leq k_j$ where $k_j < k_0$ by Lemma 2. Hence

$d_j \leq d_0$. When the robot R_j passes d_j at time $T_j \leq T + d_j \leq T + d_0$ and the target is placed right after d_j on ray r_j, then the competitive ratio for this placement of the target is given by

$$\frac{2\sum_{i\in I_j} x_i^s + x_{k_j}^s}{x_{k_j}^s} = 1 + 2\frac{\sum_{i\in I_j} x_i^s}{x_{k_j}^s},$$

according to Equation 1, for $0 \leq j \leq p-1$. The factor 2 comes from the fact that the robot has traveled to and from the origin to each turn point. Hence, the competitive ratio at time $T + d_0$ is at least

$$C_S \geq \max_{0\leq j\leq p-1}\left\{1 + 2\frac{\sum_{i\in I_j} x_i^s}{x_{k_j}^s}\right\} \geq 1 + 2\frac{\sum_{j=0}^{p-1}\sum_{i\in I_j} x_i^s}{\sum_{j=0}^{p-1} x_{k_j}^s}.$$

Here, we make use of the fact that $\max\{a/c, b/d\} \geq (a+b)/(c+d)$, for all $a, b, c, d > 0$. Note that the sum $A = \sum_{j=0}^{p-1}\sum_{i\in I_j} x_i^s$ contains as summands all x_i^s that have been explored up to time T. In particular, A includes all x_i^s that are smaller than $x_{k_0}^s$, as otherwise the robot R_0 would have explored a ray different from r_0 by Lemma 2. Similarly, there are at least $m - p + 1$ unoccupied rays at time T, one of which is r_0. These rays have each been explored to a distance $x_{l_i} \geq x_{k_0}$, for $1 \leq i \leq m - p$ since otherwise robot R_0 would have chosen one of these for exploration at time T. The smallest choice for these $m - p$ values is $x_{k_0+1}^s, \ldots, x_{k_0+m-p}^s$. Hence,

$$\sum_{j=0}^{p-1}\sum_{i\in I_j} x_i^s \geq \sum_{i=0}^{k_0+m-p} x_i^s.$$

Now consider the values d_j, for $1 \leq j \leq p-1$. The value d_j is the distance up to which ray r_j was explored before robot R_j entered it. Since robot R_j chose ray r_j and not ray r_0, Lemma 2 implies that $d_j \leq d_0 = x_{k_0}^s$. The $p-1$ largest such values are $x_{k-p+1}^s, \ldots x_{k-1}^s$ and

$$\sum_{j=1}^{p-1} d_j \leq \sum_{i=k_0-p+1}^{k_0} x_i^s.$$

Hence,

$$C_S \geq 1 + 2\frac{\sum_{j=0}^{p-1}\sum_{i\in I_j} x_i^s}{\sum_{j=0}^{p-1} x_{k_j}^s} \geq 1 + 2\frac{\sum_{i=0}^{k_0+m-p} x_i^s}{\sum_{i=k_0-p+1}^{k_0} x_i^s},$$

for all $k \geq p$. □

In order to prove a lower bound on Expression 2 we make use of the results by Gal [7] and Schuierer [21] which we state here without proof and in a simplified form for completeness. Let $G_a = (1, a, a^2, \ldots)$ be the geometric sequence in a and $X^{+i} = (x_i, x_{i+1}, \ldots)$ the suffix of sequence X starting at x_i.

Theorem 1 ([21]) *Let $X = (x_0, x_1, \ldots)$ be a sequence of positive numbers, r an integer, and $a = \overline{\lim}_{n\to\infty}(x_n)^{1/n}$, for $a \in \mathbb{R} \cup \{+\infty\}$. If F_k, $k \geq 0$, is a sequence of functionals which satisfy*

1. *$F_k(X)$ only depends on $x_0, x_1, \ldots, x_{k+r}$,*
2. *$F_k(X)$ is continuous, for all $x_i > 0$, with $0 \le i \le k+r$,*
3. *$F_k(\alpha X) = F_k(X)$, for all $\alpha > 0$,*
4. *$F_k(X+Y) \le \max(F_k(X), F_k(Y))$, and*
5. *$F_{k+i}(X) \ge F_k(X^{+i})$, for all $i \ge 1$,*

then

$$\sup_{0 \le k < \infty} F_k(X) \ge \sup_{0 \le k < \infty} F_k(G_a).$$

In particular, in our case it is easy to see that, if we set

$$F_k(X^s) = 1 + 2 \sum_{i=0}^{k+m-p} x_i^s \Big/ \sum_{i=k-p+1}^{k} x_i^s,$$

then F_k satisfies all conditions of Theorem 1. Hence,

$$C_S \;\ge\; \sup_{0 \le k < \infty} F_k(X^s) \;\ge\; \sup_{0 \le k < \infty} F_k(G_a) = \sup_{0 \le k < \infty} \left\{ 1 + 2 \sum_{i=0}^{k+m-p} a^i \Big/ \sum_{i=k-p+1}^{k} a^i \right\}.$$

Note that if $a \le 1$, then the above ratio tends to infinity as $k \to \infty$. Hence, we can assume that $a > 1$ and obtain

$$\begin{aligned} C_S \quad &\ge \quad \sup_{0 \le k < \infty} \left\{ 1 + 2 \frac{(a^{k+m-p+1} - 1)/(a-1)}{(a^{k+1} - a^{k-p+1})/(a-1)} \right\} \\ &= \quad \sup_{0 \le k < \infty} \left\{ 1 + 2 \frac{a^{k+m-p+1} - 1}{a^{k+1} - a^{k-p+1}} \right\} \\ &\overset{(a>1)}{=} \quad 1 + 2 \frac{a^{m-p}}{1 - a^{-p}} \;=\; 1 + 2 \frac{a^m}{a^p - 1}. \end{aligned}$$

The above expression is minimized for $a = (m/(m-p))^{1/p}$ and the competitive ratio is bounded from below by

$$C_S \ge 1 + 2 \frac{\left(\frac{m}{m-p}\right)^{m/p}}{\frac{m}{m-p} - 1} = 1 + 2 \left(\frac{m}{p} - 1\right) \left(\frac{m}{m-p}\right)^{m/p}.$$

Theorem 2 *There is no search strategy for a target on m rays using p robots with a competitive ratio of less than*

$$1 + 2 \left(\frac{m}{p} - 1\right) \left(\frac{m}{m-p}\right)^{m/p}.$$

Note that the above expression interpolates nicely between the various special cases that may occur. For instance, if $p = 1$, then we obtain $1 + 2\,m^m/(m-1)^{m-1}$ as previously shown [2, 7]. If there is an integer number of rays per robot, say $m = kp$ for some integer constant k, then we obtain

$$1 + 2\left(\frac{kp}{p} - 1\right)\left(\frac{kp}{kp-p}\right)^{kp/p} = 1 + 2(k-1)\frac{k^k}{(k-1)^k} = 1 + 2\frac{k^k}{(k-1)^{k-1}},$$

that is, the same competitive ratio as if each of the robots searches on a separate subset of k rays.

4 An Optimal Strategy

We now present a strategy that achieves a competitive ratio matching the lower bound we have shown above. The strategy works as follows. The robots explore the rays in a fixed cyclic order. Let $a = (m/(m-p))^{1/p}$. The sequence of return distances of the robots is given by $x_i = a^i$ for $i = 0, 1, 2, \ldots$. The kth time that robot R returns to the origin it chooses to explore ray $(kp + R) \bmod m$ up to distance x_{kp+R}. Obviously, the ith time ray r is explored, the robot explores it up to distance x_{im+r}.

So let r be a ray that is explored by robot R after it has returned the kth time to the origin. Hence, $kp + R = r \bmod m$, or equivalently $kp + R = im + r$. The total distance traveled thus far by robot R is $2\sum_{j=0}^{k-1} x_{jp+R}$. Clearly, the robot that explored ray r up to distance $x_{(i-1)m+r}$ reached the origin before robot R. Hence, r has been explored up to distance $x_{(i-1)m+r}$ when robot R travels on it and the competitive ratio in this step is given by

$$\begin{aligned} 1 + 2\frac{\sum_{j=0}^{k-1} x_{jp+R}}{x_{kp+R-m}} &= 1 + 2\frac{a^R \sum_{j=0}^{k-1}(a^p)^j}{a^R a^{kp-m}} = 1 + 2\frac{a^{kp}-1}{(a^p-1)a^{kp-m}} \\ &\le 1 + 2\frac{a^m}{a^p-1} = 1 + 2\left(\frac{m}{p} - 1\right)\left(\frac{m}{m-p}\right)^{m/p}. \end{aligned}$$

Since the bound is independent of the robot R, the ray r and the number of times the ray was visited, we obtain the following theorem.

Theorem 3 *There exists an on-line strategy for searching for a target on m rays using p robots with a competitive ratio of*

$$1 + 2\left(\frac{m}{p} - 1\right)\left(\frac{m}{m-p}\right)^{m/p}$$

which is optimal.

5 Conclusions

We present an optimal strategy for searching for a target on m concurrent rays in parallel using p robots. This strategy has a competitive ratio of

$$1 + 2\left(\frac{m}{p} - 1\right)\left(\frac{m}{m-p}\right)^{m/p}.$$

This is a generalization of the on-line construction of on-line heuristics to a distributed model. It also extends the cow path problem to multiple searchers on m concurrent rays, which has proven to be a basic primitive in the exploration of certain classes of polygons. Furthermore, it expands the field of target searching to multiple robots; a setting that more closely reflects real-world scenarios. An open problem is to generalize this algorithm to randomized or average case strategies. In similar settings, a trade-off theorem between average and worst case performance of search strategies for a single robot is known. It is natural to expect that a similar result might hold for parallel searches.

References

[1] I. Althöfer. A Symbiosis of Man and Machine Beats Grandmaster Timoshchenko. *Journal of the International Computer Chess Association.*, vol. 20, no. 1, March 1997, pp. 187ff.

[2] R. Baeza-Yates, J. Culberson, and G. Rawlins. Searching in the plane. *Information and Computation*, 106:234–252, 1993.

[3] R. Baeza-Yates and R. Schott. Parallel searching in the plane. *Comput. Geom. Theory Appl.*, 5:143–154, 1995.

[4] Margrit Betke, Ronald L. Rivest, and Mona Singh. Piecemeal learning of an unknown environment. In *Sixth ACM Conference on Computational Learning Theory (COLT 93)*, pages 277–286, July 1993.

[5] A. Datta, Ch. Hipke, and S. Schuierer. Competitive searching in polygons—beyond generalized streets. In *Proc. Sixth Annual International Symposium on Algorithms and Computation*, pages 32–41. LNCS 1004, 1995.

[6] A. Datta and Ch. Icking. Competitive searching in a generalized street. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.*, pages 175–182, 1994.

[7] S. Gal. *Search Games.* Academic Press, 1980.

[8] M. Hammar, B. Nilsson, and S. Schuierer. Parallel searching on m rays. In *Symp. on Theoretical Aspects of Compute Science*, pages 132–142. LNCS 1563, 1999.

[9] Ch. Hipke. Online-Algorithmen zur kompetitiven Suche in einfachen Polygonen. Master's thesis, Universität Freiburg, 1994.

[10] M.-Y. Kao, Y. Ma, M. Sipser, and Y. Yin. Optimal constructions of hybrid algorithms. *Proc. 5th ACM-SIAM Sympos. Discrete Algorithms*, pp. 372–381, 1994.

[11] M.-Y. Kao, J. H. Reif, and S. R. Tate. Searching in an unknown environment: An optimal randomized algorithm for the cow-path problem. In *Proc. 4th ACM-SIAM Sympos. Discrete Algorithms*, pages 441–447, 1993.

[12] D. King. Kasparov versus Deeper Blue. *Journal of the International Computer Chess Association.*, vol. 20, no. 3, September 1997, pp. 187ff.

[13] R. Klein. Walking an unknown street with bounded detour. *Comput. Geom. Theory Appl.*, 1:325–351, 1992.

[14] J. M. Kleinberg. On-line search in a simple polygon. In *Proc. of 5th ACM-SIAM Symp. on Discrete Algorithms*, pages 8–15, 1994.

[15] A. López-Ortiz. *On-line Searching on Bounded and Unbounded Domains.* PhD thesis, Department of Computer Science, University of Waterloo, 1996.

[16] A. López-Ortiz and S. Schuierer. Generalized streets revisited. In M. Serna J. Diaz, editor, *Proc. 4th European Symp. on Algorithms*, pp. 546–558. LNCS 1136, 1996.

[17] A. López-Ortiz and S. Schuierer. The ultimate strategy to search on m rays? *Theoretical Computer Science*, 261(1):267–295, 2001.

[18] A. López-Ortiz and G. Sweet. Parallel searching on a lattice. In *Proc. 13th Canadian Conference on Computational Geometry*, pages 125–128, 2001.

[19] A. Mei and Y. Igarashi. Efficient strategies for robot navigation in unknown environment. In *Proc. of 21st Intl. Colloquium on Automata, Languages and Programming*, pages 51–56, 1994.

[20] C. H. Papadimitriou and M. Yannakakis. Shortest paths without a map. In *Proc. 16th Internat. Colloq. Automata Lang. Program.*, volume 372 of *Lecture Notes in Computer Science*, pages 610–620. Springer-Verlag, 1989.

[21] S. Schuierer. Lower bounds in on-line geometric searching. *Computational Geometry: Theory and Applications*, 18(1):37–53, 2001.

[22] D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28:202–208, 1985.

[23] Y. Yin. *Teaching, Learning, and Exploration.* PhD thesis, Department of Mathematics and Laboratory for Computer Science, MIT, 1994.