ELSEVIER

Computational Geometry 5 (1995) 143–154

Computational Geometry
Theory and Applications

Parallel searching in the plane

Ricardo Baeza-Yates [a], René Schott [b,*]

[a] *Department of Computer Science, Universidad de Chile, Casilla 2777, Santiago, Chile*
[b] *CRIN, INRIA-Lorraine, Université de Nancy 1, 54506-Vandoeuvre-lès-Nancy, France*

Communicated by J.D. Boissonnat; submitted 4 January 1994; accepted 9 December 1994

Abstract

In this paper we investigate parallel searching in the plane using robots as searchers. We show that a huge number of robots are not necessary for several problems and under different assumptions. This corresponds to real situations since, actually, the number of processors of parallel machines is fixed and independent of the dimension of the problem to be solved.

1. Introduction

The problem considered in this paper was suggested by practical motion planning problems. The real stimulation came from the fact that we had the opportunity to use a parallel T-node machine with 32 processors for running our software. Therefore, we were concerned with the following natural question: Can we develop optimal parallel search in the plane with a limited number of processors (robots)? Several parallel algorithms have been developed recently in computational geometry but the complexity appears to be interesting only from a theoretical point of view since the result is usually expressed in terms of n processors ($n \to \infty$), where n is also the dimension of the problem. A recent work [6] deals with the speed-up that can be given by the parallelization of a problem when the number of processors is fixed and independent of the dimension of the problem.

We address several problems of finding a given goal in parallel with incomplete information about the goal's position. The goal is either a point in a line (one dimension) or a line in a plane (two dimensions). We model each processor as a robot, and we say that the goal is found when the robot reaches the goal. We can have several searching

* Corresponding author.

SSDI 0925-7721(95)00003-8

models. The simplest one, but not very interesting, is that when a robot finds the goal, it does not care about the other robots. A better model, which we use most of the time, is to assume that the robots have some means of instantaneous contact (say a radio) such that when a robot finds the goal all other robots stop (or go to where the goal is). Yet another model is when the robots do not have a communication device. So, there has to be some kind of protocol such that the robot that finds the goal can find the other robots and direct them to the goal. We include one example for this model.

We consider two measures: the distance travelled by the robots and the time to reach the goal. We denote by D the distance to the goal (which may be not known). Without loss of generality, we also assume that all robots are initially placed at the same starting point (origin) and that they travel at the same speed V (if you have different speeds, the worst case will always happen to the slowest robot). We call D/V a unit of time, and we use a normalized speed $V = D$ units of distance/time. The sequential solution to the problems addressed here are given in [1,2], and are related to searching without information, on-line algorithms, and motion planning [9].

The paper is organized as follows. Section 2 presents the parallel searching problem and outlines some results for searching a point on a line. Section 3 presents the bounded two dimensional case and also discusses average case analysis. Section 4 addresses the unbounded case in two dimensions. Our analysis was performed by using the Maple symbolic algebra system [5]. A preliminary version of this paper was presented at [3].

2. Searching for a point on a line

Suppose that the robot needs to find some point on a line whose position is unknown to the robot. Two different types of situations can occur:

(1) The point is at a known distance D,

(2) The point is at an unknown distance.

2.1. The point is at distance D

In the worst case, one robot travels a distance $3D$ and the time needed is equal to 3 units. The total distance travelled by two robots is $2D$ and the time necessary to find the goal is 1 unit. Obviously, more robots do not help for this task.

2.2. The point is at unknown distance

Assume that after when the point is found, the distance of it to the origin is D. In the solution for one robot a distance of $9D$ must be travelled and the time needed is equal to 9 units [1,2]. This is optimal up to lower order terms. For two robots the distance travelled is $2D$ and the time necessary to find the goal is 1 unit and more robots do not help. If both robots must reach the goal, the distance increases to $4D$ and the time to 3 units (the robot that finds the goal waits for the other robot).

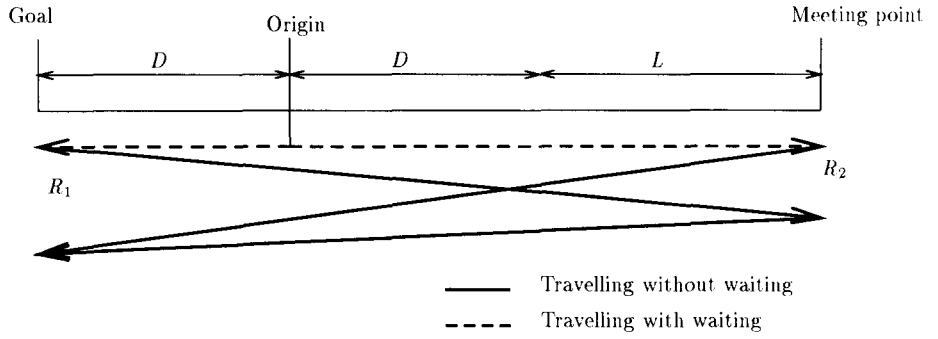

Fig. 1. Parallel solution without communication.

Let's now have the following variation: both robots cannot communicate. Hence, the robot that finds the goal must find the other robot, tell him, and then both must go back to the goal. To have a solution the robots must agree on a protocol such that they can get in contact in the future. One possibility is to stop and wait at certain intervals before any of them finds the goal (or in other words, they agree to reduce the speed). Suppose that they wait R time/distance units. The robot that finds the goal, goes back at full speed until meets the other robot and they return together (see Fig. 1). They will meet because one robot is going faster than the other robot.

After the first robot reaches the goal, we have that L is such that

$$\frac{L+2D}{V} = L\left(\frac{1}{V}+R\right),$$

where $(L+2V)/V$ is the time of traversal when at full speed by the robot that finds the goal and $L(1/V+R)$ is the time spend by the other robot after the goal is found (without this robot knowing that the goal was found). Therefore:

$$L = \frac{2D}{VR}.$$

Hence, the total distance is:

$$4D+2L+2L+4D = 8D\left(1+\frac{1}{VR}\right).$$

Therefore, asymptotically on R, the distance approaches $8D < 9D$. The total time that both robots take to find the goal is:

$$T(R) = \frac{D}{V} + DR + 2\frac{L+2D}{V} = \frac{5D}{V} + D\left(R+\frac{4}{V^2R}\right).$$

Minimizing $T(R)$ we obtain: $R = 2/V$ (they wait half the time), which gives a total distance of $12D$ and $T = 9D/V$ or 9 units of time. That is, they take the same time as one robot! In other words, the advantage of using more than one robot depends upon whether or not they have some instantaneous communication device.

Another possible protocol would be that both robots agree to return to the origin in a certain manner. However, this idea is not better than the one just analyzed on distance or time. However, if we have more robots, we can have two robots searching and coming back to certain points, while the other robots can carry messages between the searchers until the goal is found. In this case, the goal can be reached in a smaller time.

The average case analysis depends on the probability distribution of finding the goal. In [1,2] it is shown that the optimal algorithm for many centered distributions over a finite range has an *infinite* number of turns. On the other hand, for a uniform distribution, the optimal algorithm is the same as the optimal worst case algorithm. With two robots, the optimal worst case solution would change only if the probability distribution of finding the goal is not symmetric with respect to the starting point.

3. Searching for a line at unit distance

We are looking for the set of paths of the robots, all starting at origin O, with smallest overall length, such that the convex hull of the union of these paths contains the circle of radius D centered at O. Searching for a line of arbitrary slope a known distance away (say 1 kilometer) in the plane was posed by Bellman [4] and was solved by Isbell [8]. The optimal worst case distance walked by a single robot is $1 + 7\pi/6 + \sqrt{3} \approx 6.39$. For two robots, we show that the asymmetric algorithm presented is better than a symmetric one.

The algorithm is as follows. One goes straight with an angle $(x + y)$ of the other for a distance of length $1/\cos(y)$, the other goes straight for a distance of length $1/\cos(x)$, then goes to the circle, follows it and takes the tangent perpendicular to the last tangent visited by the first robot (see Fig. 2)

The overall distance is:

$$\frac{1}{\cos(y)} + \frac{1}{\cos(x)} + \tan(x) + \frac{3\pi}{2} - 2(x+y) + 1.$$

This is optimized for

$$x = \frac{\pi}{6}(30°), \quad \text{and} \quad y = \arcsin\left(\frac{\sqrt{17}-1}{4}\right) \approx 0.8959\ (51.3°)$$

giving $d(2) = 6.206$ (slightly better than one robot). The total time is:

$$t = \frac{1}{\cos(x)} + \tan(x) + \frac{3\pi}{2} - 2(x+y) + 1.$$

For the above values of x and y we obtain $t = 4.605$, which is about 72% of the time for the one robot solution.

However, this asymmetric algorithm is not time optimal. The optimal is achieved when both robots travel the same distance. This implies the same value of x, but we have $y \approx 1.303$ (74.7°), giving $t = 3.7898$ which is only 59% of the one robot solution.

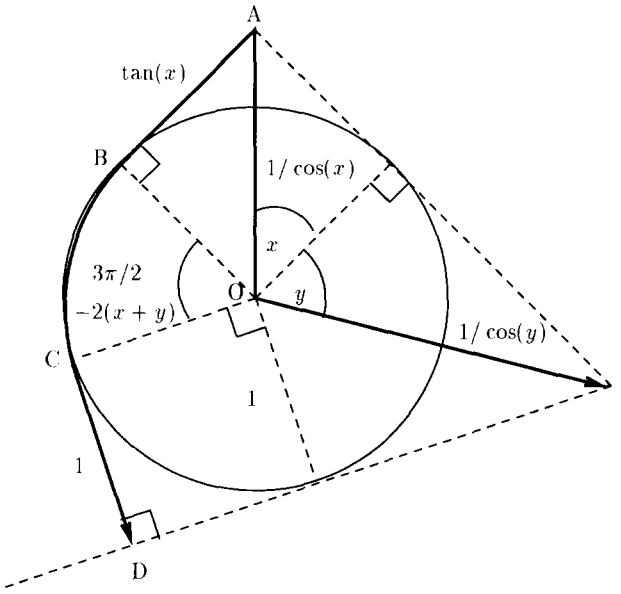

Fig. 2. Asymmetric algorithm for two robots.

However, the total distance jumps to 7.579, that is, 18% more than the one robot solution.

The total worst case distance of a symmetric algorithm for two robots (see Fig. 3) is given by

$$d = 2\left(\frac{1}{\cos(x)} + \tan(x) + \frac{\pi}{2} - 2x + 1\right)$$

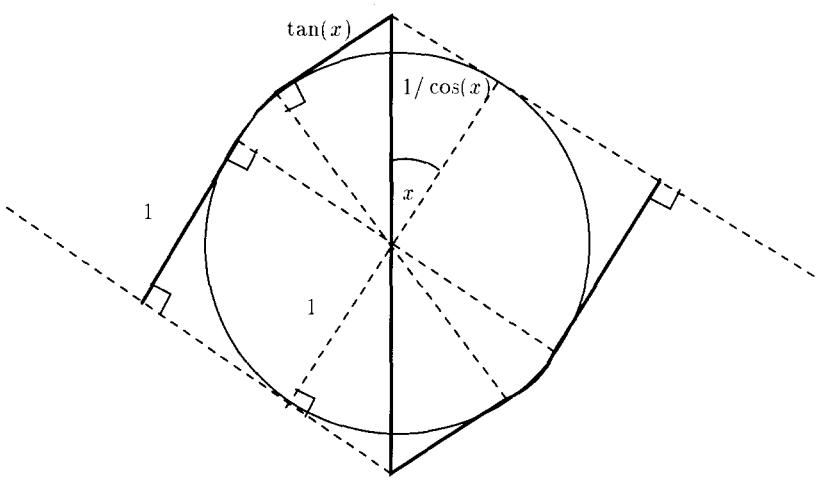

Fig. 3. Symmetric algorithm for two robots.

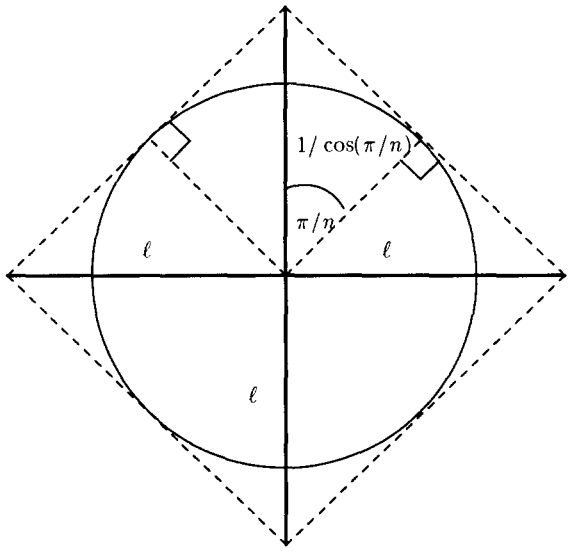

Fig. 4. Symmetric algorithm for $n > 2$ shown for $n = 4$.

which is also optimized for $x = \pi/6$ giving a total distance of $2(1 + \sqrt{3}) + \pi/3 \approx 6.511$ but a time of 3.2556 which is about 51% of the one robot solution.

For $n = 3$ or more robots, we found no better algorithm than the symmetric one: they all go straight with an angle of $2\pi/n$ between two neighbouring robots. They stop after travelling $\ell = 1/\cos(\pi/n)$ (see Fig. 4). The total distance for n robots is:

$$d(n) = \frac{n}{\cos\left(\frac{\pi}{n}\right)}, \qquad \text{for } n > 2.$$

Table 1 shows the total distance in function of n, including the time consumed. Note that there is a trade-off between distance (optimal for $n = 4$) and time (optimal for large n). The obvious lower bound for the distance is $\max(d(1)/n, n)$ considering that the n

Table 1
Distance and time for our best algorithm in function of n.

n	distance	lower bound	time	lower bound
1	6.3972	6.3972	6.3972	6.3972
2	6.2059	3.1986	3.2556	1.5993
3	6.0	3.0	2.0	1.0
4	5.6569	4	1.4142	1.0
5	6.1803	5	1.2361	1.0
6	6.9282	6	1.1547	1.0
7	7.7694	7	1.1099	1.0
8	8.6591	8	1.0824	1.0
9	9.5776	9	1.0642	1.0
10	10.5146	10	1.0515	1.0
$n \to \infty$	n	n	1	1.0

robots cannot do better than one robot divided by the number of robots, and that all the robots should get to the circle to be able to find the goal (and all of them should be used). The lower bound for the total time is then $\max(d(1)/n^2, 1)$. These lower bounds are also shown in Table 1.

For the non-communicating robots model, it is enough for the robot that finds the goal to follow a circle of radius ℓ to tell the other robots where the goal is. The extra distance travelled by all the robots to reach the goal is:

$$\frac{\pi(n+1)}{\cos\left(\frac{\pi}{n}\right)}$$

which is also minimized for $n = 4$.

3.1. The average case analysis

We first examine the case of two robots using an asymmetric algorithm. Then we compare with the symmetric algorithm. A general result is also obtained for n robots using the symmetric algorithm. There are no known lower bounds for this problem.

3.1.1. The asymmetric algorithm for two robots

The asymmetric algorithm is described in Fig. 2. We first compute the average total distance. We assume that $y \geq x$, $1/\cos(y) \leq 1/\cos(x) + \tan(x)$, and $3\pi/2 - 2x - 2y \geq 0$, because in that range we have the optimal value for x and y. The analysis for the other cases is similar.

The total distance is explained below in detail:

- Both robots travel at the same time covering all the tangents from $-x$ to x for each path. Thus, the average distance until the first robot reaches A is:

$$2\int_{-x}^{x} \frac{1}{\cos(\theta)} \frac{d\theta}{2\pi}.$$

- The second robot continues the search covering the tangents from x to y and $-x$ to $-y$. The first robot starts travelling from A to B. Therefore, considering that $1/\cos(\theta)$ is an even function, we have

$$4\int_{x}^{y} \frac{1}{\cos(\theta)} \frac{d\theta}{2\pi}.$$

- After the second robot stops, the first robot will complete the AB segment if the line has not been already found. After that, it follows the BC arc. Then, we have

$$\frac{2\pi - 2x - 2y}{2\pi}\left(\frac{1}{\cos(x)} + \tan(x) + \frac{1}{\cos(y)}\right) + \int_{0}^{\frac{3\pi}{2} - 2x - 2y} \theta \frac{d\theta}{2\pi}.$$

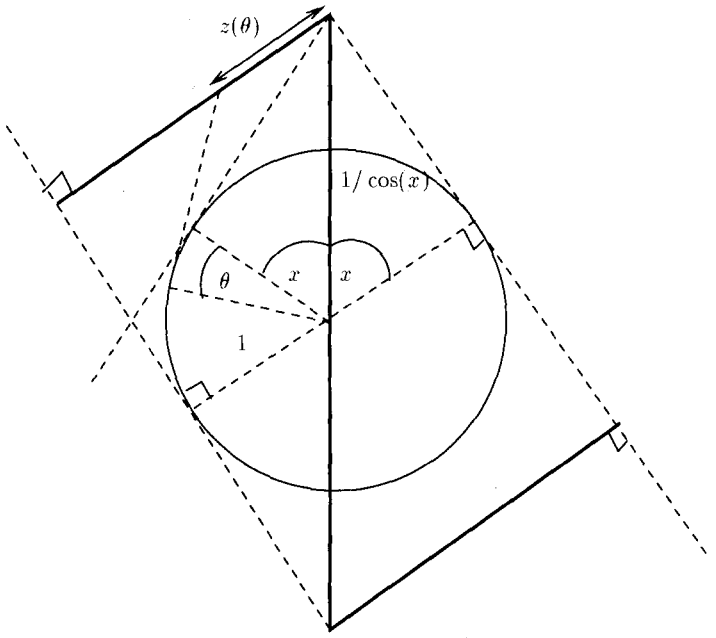

Fig. 5. Another symmetric algorithm for two robots.

• After traveling the BC arc, if the line is not found, we have the CD segment. The additional average distance for this case is:

$$\frac{\pi/2}{2\pi}\left(\frac{3\pi}{2} - 2x - 2y\right) + \int_0^{\frac{\pi}{2}} \tan\left(\frac{\theta}{2}\right)\frac{d\theta}{2\pi}.$$

Adding all these contributions we obtain the total distance. This distance is optimized for $x = 1.0573\cdots$ $(60.6°)$ and $y = 1.0641\cdots$ $(61.0°)$ giving $d = 3.01156\cdots$.

The total average time is given by the distance travelled by the first robot. Using the values obtained for x and y we get $t = 1.9122\cdots$. However, as before this is not time minimal. The optimality on t is achieved for $x = 1.014\cdots$ $(58.1°)$ and $y = 1.370\cdots$ $(78.5°)$, giving $t = 1.671$ and $d = 3.613$.

3.1.2. Symmetric case for two robots

The symmetric algorithm shown in Fig. 3 is not the best on average. In fact, the algorithm shown in Fig. 5 is better. This happens because going further away from the perimeter of the circle maximizes the chance of finding the goal earlier.

Let d be the total average distance travelled by the two robots. The distance d is given by:

$$d = 2\left(\int_{-x}^{x} \frac{1}{\cos(\theta)}\frac{d\theta}{2\pi} + \frac{2\pi - 4x}{2\pi\cos(x)} + \int_0^{\pi-2x} z(\theta)\frac{d\theta}{2\pi}\right)$$

Table 2
Results for the symmetric algorithm

n	d	t
1	3.4699[7]	3.4699
2	2.5392	1.2696
3	3.2576	1.0859
4	4.1222	1.0306
5	5.0732	1.0146
6	6.0492	1.0082
7	7.0354	1.0051
8	8.0267	1.0033
9	9.0209	1.0023
10	10.017	1.0017
$n \to \infty$	n	1

with $z(\theta) = (\cos(\theta + x)/\cos(x) - 1)/\cos(\theta + 2x)$. This function is optimized for $x = 1.0095 \cdots$ $(57.8°)$ giving $d = 2.539$, which is a clear improvement over the asymmetric algorithm. The expected time t is $d/2$.

3.1.3. The symmetric algorithm for n robots

We consider n robots starting from the center of the circle as in Fig. 4. The angle between two adjacent robots is now $2\pi/n$. When a robot R finds the line, the other have walked $n - 1$ times the distance travelled by R. The probability that none of the robots have found the line when a robot finds it, in function of the angle covered is

$$p(\theta) = 1 - \frac{n\theta}{\pi}.$$

That average distance walked for that robot is then

$$d_1 = 1 + \int_{-\frac{\pi}{n}}^{\frac{\pi}{n}} p(\theta)\left(\frac{1}{\cos\theta} - 1\right)\frac{d\theta}{2\pi} = 1 - \frac{1}{n} + \frac{1}{\pi}\ln\left(\tan\left(\frac{\pi}{4} + \frac{\pi}{2n}\right)\right).$$

With n robots, the average (total) distance travelled is therefore given by:

$$d = nd_1 = n - 1 + \frac{n}{\pi}\ln\left(\tan\left(\frac{\pi}{4} + \frac{\pi}{2n}\right)\right) = n + \frac{\pi^2}{6n^2} + O(n^{-4}).$$

Numerical evaluation with Maple gives the results shown in Table 2, giving for completeness also the values for $n < 3$.

Of course the average time needed by n robots ($t = d/n$) searching with the symmetric algorithm is easily deduced thanks to the results above, and are given in Table 2. As we can see, for example, for $n = 10$ the parallel search of a line at unit distance is very close to the limit 1.

4. Searching for a line at an arbitrary distance

When the line is at an unbounded distance the best known algorithm for one robot is to follow a logarithmic spiral with radius $r(\theta) = k^{\theta}$ with $k \approx 1.237$, which gives a worst case asymptotic ratio between the distance travelled to the line distance of 13.81 [1,2].

For $n > 2$ robots we can use the symmetric algorithm of the previous section, obtaining the same results, because they do not depend on the distance to the line. So the distance travelled goes from 13.81 for $n = 1$ to 6 for $n = 3$. What about $n = 2$? The algorithms in the previous section for this case used the fact that the line distance was known. Based on the spiral search, a simple algorithm for $n = 2$ is to use two logarithmic spirals, with an angle difference of π, that is one robot follows the polar curve $r_1(\theta) = x^{\theta}$ while the other robot uses $r_2(\theta) = x^{\theta+\pi}$. The worst case happens when the line is tangent to r_1 and missed by it, and then is found by the other robot. Using the logarithmic spiral property that a tangent to it has angle ϕ given by the relation $x = \exp(\cot \phi)$, it is possible to show that if the tangent is missed at the angle α by the first robot, then it is found by the second robot in the angle $\alpha + \pi - \beta$, where

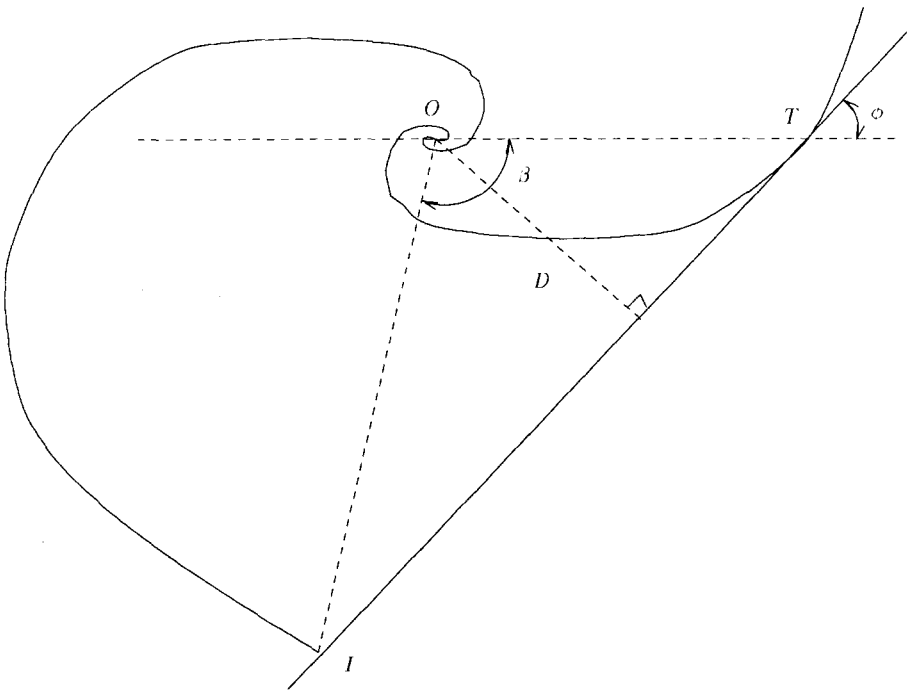

Fig. 6. Two robots algorithm for a line at unbounded distance.

β depends on ϕ and is obtained from the law of sines in the triangle formed by the origin O, the tangent point T and the intersection point I (see Fig. 6). That is

$$\frac{r_1(\alpha)}{r_2(\alpha+\pi-\beta)} = x^{\pi-\beta} = \frac{\sin(\phi)}{\sin(\pi-\phi-\beta)}.$$

The worst case asymptotic ratio is given by

$$\frac{2(\text{distance travelled from } O \text{ to } T)}{D}.$$

The distance travelled from O to T is

$$\int_{-\infty}^{\alpha+\pi-\beta} x^{\theta} d\theta = r_1(\alpha+\pi-\beta)\sec(\phi)$$

and the distance to the line, D, is given by

$$D = r_1(\alpha)\sin(\phi).$$

So, the worst case ratio is $x^{\pi-\beta}\sin^2(\phi)$. Minimizing this equation and the equation for β in function of $x(\phi)$ we obtain a non-linear system of two equations where x and β can be computed numerically obtaining $x = 1.9088\cdots$ (or $\phi \approx 57°$) and $\beta = 1.7872\cdots$ ($\approx 102°$) which gives a worst case ratio of $10.5288\cdots$ using 5.2644 units of time. If the first robot has to meet the second robot at the point where the goal was found, we have to add $5.7161\cdots$ units of time. If the second robot just has to reach the line, only 2 units of time are needed (uses a perpendicular to the line).

5. Conclusion and further aspects

We have shown that:

- searching in parallel for a point on a line at known or unknown distance has similar complexity,
- searching in parallel for a line at unit distance can be done efficiently using a limited number n of robots (that is, processors). For this problem it appears that the asymmetric algorithm presented here is better than a symmetric one for $n = 2$ in the worst case, but not on the average. For $n > 2$, the best is a symmetric algorithm.
- the distance of the line is not relevant for $n > 2$, and that there is a clear gap between using one or more than one robot when searching in the plane.
- robot communication is necessary to achieve efficient parallelism, because otherwise meeting protocols have to be devised.

We have seen that as the number of robots increases, the searching algorithm is simpler, when $n > 2$. However, the problem complexity seems to be more complex for $n = 2$. In general, if we have d dimensions and we are trying to find a goal that is a $(d-1)$-dimensional hyperplane, it is enough to have $d+1$ robots to be able to use a simple symmetric algorithm. For example, in three dimensions, we need 4 robots whose positions at each moment of searching form the vertices of a regular tetrahedron.

For two dimensions, the optimal algorithm for one robot when the distance of the goal is not known seems to be a particular logarithmic spiral [1,2]. For $n > 2$ we can use a symmetric algorithm. For $n = 2$ we have presented a symmetric algorithm that improves over the non-parallel solution, but not by much.

We have not shown any lower bounds, besides the obvious ones. It is not clear which are the adequate techniques for this task. In [1,2] the lower bounds presented were obtained using several different arguments.

Finally, we have been searching for a $(d-1)$-dimensional goal on a d dimensional space with a 0-dimensional object (robot). Another possible generalization is to find the goal with an i-dimensional object, for $0 \leq i \leq d-2 (d > 1)$. Another variation is to change the metric space (for example, a Manhattan metric).

Acknowledgements

The first author thanks the helpful comments of Riccardo Ferrari. The second author thanks the Department of Computer Science, Universidad de Chile, Santiago, for kind hospitality. The authors are also grateful to Laurent Alonso for checking carefully the calculations and to J.D. Boissonnat for several suggestions. The pertinent remarks of an anonymous referee led to substantial improvement of the contents and the presentation.

References

[1] R.A. Baeza-Yates, J. Culberson and G. Rawlins, Searching with uncertainty, in: R. Karlsson and A. Lingas, eds., Proceedings SWAT 88, First Scandinavian Workshop on Algorithm Theory, Lecture Notes in Computer Science 318 Halmstad, Sweden (1988) 176–189.

[2] R.A. Baeza-Yates, J. Culberson and G. Rawlins, Searching in the plane, Inform. Comput. 106 (1993) 234–252.

[3] R. Baeza-Yates and R. Schott, Parallel searching in the plane, XII Int. Conf. of the Chilean Computer Society, Santiago, Chile (1992) 269–279.

[4] R. Bellman, A minimization problem, Bulletin AMS 62 (1956) 270.

[5] B. Char, G. Geddes, G. Gonnet, B. Leong, M. Monagan and S. Watt, MAPLE V Language and Library Reference Manual (Springer, Berlin, 1991).

[6] C. Delporte-Gallet, H. Fauconnier and M. Nivat, Parallélisation d'algorithmes avec un nombre fixe de processeurs, Theoret. Comput. Sci. 24 (353–386) 1990.

[7] B. Gluss, An alternative solution to the lost at sea problem, Naval Research Logistics Quarterly 4 (1961) 117–121.

[8] J.R. Isbell, An optimal search pattern, Naval Research Logistics Quaterly 4 (1957) 357–359.

[9] C. Papadimitriou and M. Yannakakis, Searching without a map, Proceed. ICALP'89, Stresa, Italy, LNCS 372 (1989) 610–620.