
Competitive Disconnection Detection in On-Line Mobile Robot Navigation

Yoav Gabriely¹ and Elon Rimon²

¹ Technion, Israel Institute of Technology yoavga@tx.technion.ac.il

² Technion, Israel Institute of Technology elon@robby.technion.ac.il

Abstract *This paper is concerned with target unreachability detection during on-line mobile robot navigation in an unknown planar environment. Traditionally, competitiveness characterizes an on-line navigation algorithm in cases where the target is reachable from the robot's start position. This paper introduces a complementary notion of competitiveness which characterizes an on-line navigation algorithm in cases where the target is unreachable. The disconnection competitiveness of an on-line navigation algorithm measures the path length it generates in order to conclude target unreachability relative to the shortest off-line path that proves target unreachability from the same start position. It is shown that only competitive navigation algorithms can possess disconnection competitiveness. A competitive on-line navigation algorithm for a disc-shaped mobile robot, called CBUG, is described. This algorithm has a quadratic competitive performance, which is also the best achievable performance over all on-line navigation algorithms. The disconnection competitiveness of CBUG is analyzed and shown to be quadratic in the length of the shortest off-line disconnection path. Moreover, it is shown that quadratic disconnection competitiveness is the best achievable performance over all on-line navigation algorithms. Thus CBUG achieves optimal competitiveness both in terms of connection and disconnection paths. Examples illustrate the usefulness of a doubly competitive algorithm in terms of path stability.*

1 Introduction

This paper is concerned with target unreachability detection during mobile robot navigation in a planar environment populated by unknown obstacles. The robot has no a priori information about the environment, but may locally acquire this information using its on-board sensors. This class of on-line problems has a wide range of applications. Examples are navigation to various targets for mail and material delivery in offices and factories, and planetary exploration and sample acquisition. The most critical parameter in such tasks

is physical travel time rather than on-board computation time. Under a uniform velocity assumption travel time corresponds to path length. Hence in this paper navigation algorithms are classified in terms of length of the path traveled by the robot during algorithm execution. Before discussing target unreachability detection, we summarize the relevant literature.

Mobile robot on-line algorithms are discussed in the robotics and computational geometry literature. Roboticians usually emphasize the type of sensors required to achieve a given task, and the problems considered here are referred to as sensor based motion planning [6, 7]. Notable early papers in this area describe the algorithms *BUG1/BUG2* [17] and *ALG1/ALG2* [20] for navigating a two degrees-of-freedom mobile robot in an unknown planar environment using position and tactile sensors. These works have been extended to navigation in planar environments using vision and laser sensors [15, 16, 18, 21]. However, the performance of these algorithms is typically characterized in terms of geometric parameters of the environment such as total obstacle perimeter, without any reference to the length of the optimal off-line solution, denoted l_{opt} . As a result, these algorithms may be fooled to generate lengthy paths in situations where the optimal off-line path is very short.

Computational geometry researchers introduced the following notion of competitiveness. An algorithm for a task P is said to be *competitive* if its solution to every instance of P is bounded by a constant times the optimal off-line solution. An early influential paper investigates navigation of a point robot in an unknown planar environment consisting of m radial corridors [1]. (This problem has its origin with a simpler problem, where a cow seeks an entry to a pasture along an unknown fence which corresponds to $m = 2$ corridors [3].) However, a point robot cannot achieve any form of competitive navigation in general environments [1, 14, 19]. Subsequent papers discuss on-line navigation of point robots in specific classes of rectangular rooms [4, 5, 10], rooms with square-shaped obstacles [19], and generalized streets [8, 14]. All of these papers strive to achieve *linear* competitiveness (i.e. path length bounded by a constant times l_{opt}) in specific classes of environments.

In contrast, we depart from the point robot paradigm and assume that the robot is a disc of physical size $D > 0$. While this assumption may seem obvious, only few papers make use of this assumption (e.g. [9]). We have recently reported on *CBUG*, an on-line navigation algorithm for a size D robot moving in a general planar environment [13]. This algorithm generates a path whose length is bounded from above by a *quadratic* function of l_{opt} . Moreover, we have shown that any on-line navigation algorithm generates in worst case a path whose length is bounded from below by a quadratic function of l_{opt} . Hence the quadratic bound of *CBUG* is tight.

This paper focuses on the performance of on-line navigation algorithms in cases where the target is unreachable from the robot's start position. This notion is known as *disconnection proofs* in off-line graph search algorithms. In the motion planning literature, disconnection proofs appear in the context of random path planning, where an off-line sampling technique detects

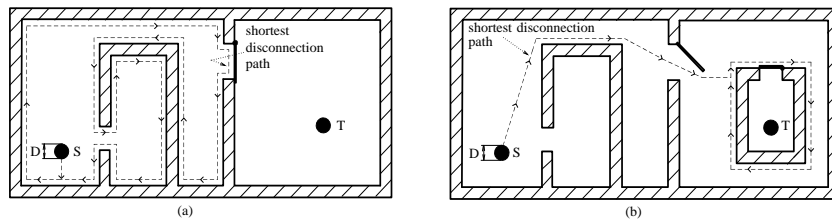


Fig. 1. All disconnection paths from S trace an obstacle boundary surrounding either S or T .

target unreachability of a polyhedral robot [2]. The following on-line version of a disconnection proof is a contribution of this paper. Let a *disconnection path* be a path that starts at S and proves that a target T is unreachable (Figure 1). Let λ be length of the disconnection path generated from S by an on-line algorithm. Let λ_{opt} be length of the shortest off-line disconnection path which starts at the same S . Then an algorithm has a $h(\lambda_{opt})$ *disconnection competitiveness* if $\lambda \leq h(\lambda_{opt})$ in all instances where T is unreachable from S . Based on this definition, we first show that a navigation algorithm must be competitive in order to be disconnection competitive. Then we establish that *CBUG* generates a disconnection path whose length is bounded from above by a *quadratic* function of λ_{opt} . Finally, we establish that any navigation algorithm in an unknown planar environment generates in worst case a disconnection path whose length is bounded from below by a quadratic function of λ_{opt} . The quadratic disconnection bound of *CBUG* is thus tight.

The structure of the paper is as follows. In the next section we define generalized competitiveness and introduce the notion of disconnection competitiveness. The *CBUG* algorithm is reviewed in Section 3. Its quadratic upper bound is summarized and shown to match the universal lower bound over all on-line navigation algorithms. The disconnection competitiveness of *CBUG* is analyzed in Section 4. It is shown that the length of the disconnection path traveled by the robot during execution of *CBUG* is at most quadratic in λ_{opt} . It is also shown that any on-line navigation algorithm generates in worst case a disconnection path whose length is at least quadratic in λ_{opt} , implying that up to constants *CBUG* has optimal disconnection competitiveness. Section 5 discusses the effect of double competitiveness on path stability, and compares the performance of *CBUG* relative to non-competitive algorithms. The concluding section mentions several open problems.

2 Definition of Disconnection Competitiveness

This section describes our basic setup, then proceeds with formal definitions of connection and disconnection competitiveness. Our basic assumptions are as follows. We assume a planar unknown environment populated by stationary

and compact obstacles. The mobile robot is a freely moving planar disc of size $D > 0$, where D is a given constant. The robot is equipped with two sensors which are assumed ideal. The first sensor measures the robot's position with respect to a fixed reference frame. The second is an obstacle detection tactile sensor which allows tracing of an obstacle boundary. In addition to sensors the robot has on-board memory in which information on the environment can be accumulated.

Next consider the parameters governing the performance of mobile robot tasks. The three most significant parameters are physical travel time, on-board computation time, and on-board memory. In order to simplify the ensuing analysis, we associate physical travel time with length l of the path traveled by the robot. As for on-board computation time, we limit our discussion to algorithms that take *polynomial time* to compute each physical motion step of the robot. Since the time required for a physical motion step is typically several orders of magnitudes longer than the execution time of an on-board computation step, we focus on l as the main performance parameter. Last, we limit the discussion to algorithms whose storage requirement is at most *linear* in the size of the environment.

Thus l denotes length of the path traveled by the robot, while l_{opt} denotes length of the optimal off-line path. The following definition generalizes the traditional notion of linear competitiveness to any functional relationship between l and l_{opt} .

Definition 1 (connection competitiveness) *An on-line navigation algorithm is $\mathbf{f}(l_{opt})$ competitive when its path length l is bounded from above by a scalable function $f(l_{opt})$ over all instances where the target is reachable. In particular, $l \leq c_1 l_{opt} + c_0$ is the traditional linear competitiveness, while $l \leq c_2 l_{opt}^2 + c_1 l_{opt} + c_0$ is quadratic competitiveness, where the c_i 's are positive constants that depend on the robot size D .*

The meaning of *scalability* is as follows. When performance is measured in physical units such as meters m , one must ensure that both sides of the relationship $l \leq f(l_{opt})$ possess the same units, so that change of scale would not affect the bound. For instance, the coefficient c_2 in the relationship $l \leq c_2 l_{opt}^2 + c_1 l_{opt} + c_0$ must have units of m^{-1} , c_1 must be unitless, and c_0 must have units of m . Note that the definition of competitiveness focuses on a particular navigation algorithm. However, our objective is to characterize the least upper bound that can be achieved over all on-line navigation algorithms. This objective requires the following definition of a universal lower bound.

Definition 2 A universal lower bound *on the competitiveness of on-line navigation is a lower bound $g(l_{opt})$ such that $l \geq g(l_{opt})$ over all on-line navigation algorithms for this task.*

Note that the universal lower bound characterizes the on-line navigation task itself, not any specific algorithm for this task. When the competitive upper

bound of a specific algorithm matches the universal lower bound up to constants, the bound itself becomes the *competitive complexity class* of the task [12]. Let us now define disconnection competitiveness. Recall that λ denotes length of the path traveled by the robot from a start S until it halts with a conclusion that the target T is unreachable. Recall, too, that λ_{opt} denotes length of the shortest off-line path which starts at the same S and proves that T is unreachable. The following definition is analogous to the definition of connection competitiveness.

Definition 3 (disconnection competitiveness) *An on-line navigation algorithm is $h(\lambda_{opt})$ disconnection competitive when its path length λ is bounded from above by a scalable function $h(\lambda_{opt})$ over all instances where the target is not reachable. In particular, $\lambda \leq c_2\lambda_{opt}^2 + c_1\lambda_{opt} + c_0$ is quadratic disconnection competitiveness, where the c_i 's are positive constants that depend on the robot size D .*

The last definition concerns the least upper bound on disconnection competitiveness.

Definition 4 A universal lower bound *on the disconnection competitiveness of on-line navigation is a lower bound $e(\lambda_{opt})$ such that $\lambda \geq e(\lambda_{opt})$ over all on-line navigation algorithms for this task.*

Note that here, too, the universal lower bound is not associated with a specific on-line algorithm, but rather characterizes the on-line navigation task itself.

3 The CBUG Algorithm

For clarity of presentation, we describe the algorithm for a point robot equipped with position and touch sensors, moving in a planar environment populated by unknown obstacles. The point robot represents the configuration of the disc robot, and the “obstacles” are c-space obstacles induced from the physical ones. The principle idea of *CBUG* is as follows. Given a start S and target T , the robot selects an initial ellipse with focal points S and T and area A_0 , and searches for T in the portion of the ellipse accessible from S . The search is executed with the classical *BUG1* algorithm reviewed below, which regards the bounding ellipse as a virtual obstacle³. If the target is detected the algorithm terminates. Otherwise the robot repeats the process in ellipses with areas $2^i A_0$ for $i = 1, 2, \dots$ until the target is found or determined to be inaccessible from S (Figure 2). The basic algorithm treats the bounding ellipse as an obstacle whose boundary must be traced by the robot. A more advanced

³ Other sub-algorithms such as *ALG1* [20] can be used, but the principle bounds reported here would remain the same. Simulations of *CBUG* with *BUG1* and *ALG1* as sub-algorithms are discussed below.

version of the algorithm described below does not require any tracing of the bounding ellipse. A description of the basic algorithm follows.

Basic CBUG Algorithm:

Sensors: Position and tactile sensors.

Input: A start S , a target T , an initial ellipse with focal points S and T and area A_0 .

Initialization: Set $S_1 = S$. Set initial search area $A(1) = A_0$. Set $i = 1$.

Repeat

1. Starting at S_i , search for T using *BUG1* in ellipse of area $A(i)$ with focal points S, T .
 2. If *BUG1* terminates at T : STOP, target is found.
 3. If *BUG1* determines that an obstacle boundary separates S from T (see text):
 - 3.1 If obstacle boundary does not intersect i^{th} bounding ellipse: STOP, target is unreachable.
 - 3.2 Set S_{i+1} at point where *BUG1* terminated (see text).
 4. Set $A(i+1) = 2A(i)$. Set $i = i + 1$.
- (End of repeat loop)

First let us review the *BUG1* sub-algorithm. Under *BUG1* the robot moves from the i^{th} start position towards the target until it hits an obstacle. Then it circumnavigates the obstacle in a clockwise direction while recording the closest point to the target along the current boundary as p_{min} . When the obstacle circumnavigation is complete, the robot returns to p_{min} along the shorter boundary segment. If the direction from p_{min} to T points into the current obstacle, the obstacle necessarily separates S from T and the target is unreachable [17]. Otherwise the robot resumes its motion to the target until the next obstacle is encountered or the target is found.

Based on the assumption of compact obstacles, the bounding ellipses of *CBUG* eventually contain a path to the target if one exists, or contain an entire obstacle boundary which separates the start from the target. At this stage the *BUG1* sub-algorithm finds a path to the target if one exists or determines target unreachability. Note that *CBUG* determines target unreachability only when the separating obstacle boundary is wholly physical and does not contain portions of the bounding ellipse. Also note that *CBUG* requires *constant memory*: S and T , the current obstacle hit point, the current p_{min} , distances along the current obstacle boundary, and the current search area $A(i)$.

Example 1. Consider the execution of *CBUG* in the office-like environment shown in Figure 2(a). Starting at S , the disc robot determines that the initial ellipse blocks its path to T (Figure 2(b)). Hence it doubles the ellipse's area and resumes the search from the point which was closest to T , marked as S_2 in Figure 2(c). The robot next determines that the new bounding ellipse still blocks its path to T . Hence it doubles the ellipse's area for the second time

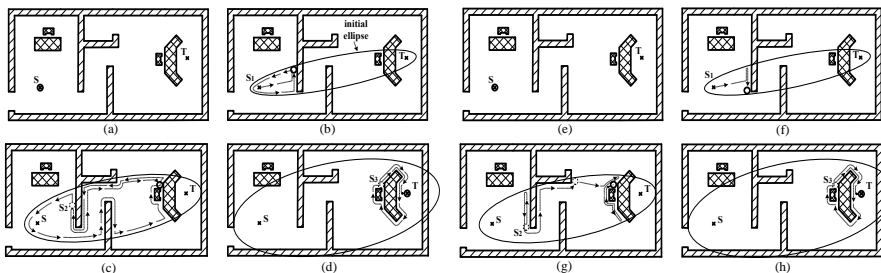


Fig. 2. (a)-(d) Execution example of the basic *CBUG*. (e)-(h) The modified *CBUG* does not require tracing of the bounding ellipses.

and resumes the search at the point S_3 (Figure 2(d)). This last search ends successfully at the target.

Next we describe a practical speedup of *CBUG* that eliminates the need to trace the bounding ellipses. When the robot hits an obstacle, either the entire obstacle boundary is contained inside the current bounding ellipse, or the boundary segment which contains the hit point has its two endpoints on the bounding ellipse. The modified algorithm requires that the robot trace an obstacle boundary until one of two events happens. Either the robot circumnavigates the entire obstacle boundary, or it reaches an endpoint of the boundary on the current bounding ellipse. In the latter case the robot *reverses* its boundary tracing direction and continues along the obstacle boundary until reaching the other endpoint of the boundary segment. From this stage onward the algorithm resumes execution of the sub-algorithm *BUG1*.

Example 2. An execution of the modified *CBUG* on the same office-like environment is shown in Figure 2(e)-(h). Each time the robot hits an obstacle, it initiates a clockwise circumnavigation of the obstacle boundary. In the first and second stages the robot encounters during boundary tracing the current bounding ellipse (Figure 2(f)-(g)). The robot consequently reverses its tracing direction until the other endpoint of the boundary segment is encountered. In both stages the path taken by the robot is significantly shorter than the path taken under the basic algorithm.

The following result asserts that the path generated by *CBUG* to an accessible target is bounded by a quadratic function of l_{opt} .

Proposition 3.1 ([13]) *If T is reachable from S , the basic *CBUG* algorithm finds the target using a path of length l satisfying the upper bound,*

$$l \leq \frac{6\pi}{D} l_{opt}^2 + \|S-T\| + \frac{6A_0}{D}, \quad (1)$$

where l_{opt} is length of the shortest off-line path from S to T , D is the disc-robot size, and A_0 is area of the initial ellipse.

Note that the three summands in (1) have length units, so the upper bound is scalable. The next result asserts that the universal lower bound on connection competitiveness is also quadratic in l_{opt} .

Theorem 1 ([13]) *Any navigation algorithm in an unknown planar environment to a reachable target generates in worst case a path of length l satisfying the quadratic lower bound,*

$$l \geq \frac{4\pi}{3(1+\pi)^2 D} (1-\epsilon) l_{opt}^2, \quad (2)$$

where l_{opt} is length of the shortest off-line path from S to T , D is the disc-robot size, and $\epsilon > 0$ is an arbitrary small constant.

The theorem implies that the connection competitiveness of *CBUG* is tight. The on-line navigation of a disc robot in planar environments thus belongs to the *quadratic competitive complexity class*. Note that (1) and (2) approach infinity as D approaches zero. This is consistent with earlier observations that a point robot cannot achieve any form of competitive on-line navigation in general planar environments [1, 14, 19].

4 Disconnection Analysis of CBUG Algorithm

This section begins with a generic assertion that connection competitiveness is necessary for disconnection competitiveness. Then we derive an upper bound on the disconnection competitiveness of the basic *CBUG* algorithm. Finally we derive a universal lower bound on disconnection competitiveness.

Proposition 4.1 *If an on-line navigation algorithm possess an upper bound $h(\lambda_{opt})$ on its disconnection competitiveness, it possesses an upper bound $f(l_{opt})$ on its connection competitiveness.*

Proof sketch: Consider a scenario where T can be reached from S . Let us assume that T can be surrounded by a small disc of radius δ which is free from obstacles. Let us further assume that the on-line algorithm guides the robot directly to the target within this small disc. We now render the target inaccessible by surrounding it with a disc-obstacle of radius δ . In this case the shortest disconnection path consists of the shortest path from S to the disc, and a loop around the disc. The length of the shortest disconnection path is $\lambda_{opt} = l_{opt} + \epsilon$, where l_{opt} is the original shortest off-line path from S to T and $\epsilon = 2\pi\delta - \delta$. By assumption any on-line disconnection path has length λ satisfying $\lambda \leq h(\lambda_{opt})$. Any disconnection path must circumnavigate the small disc surrounding T . Hence it can be converted to a path from S to T of length $l \leq h(\lambda_{opt}) - \epsilon$. Substituting $\lambda_{opt} = l_{opt} + \epsilon$ gives the upper bound $l \leq f(l_{opt})$, where $f(l_{opt}) = h(l_{opt} + \epsilon) - \epsilon$. \square

In the following analysis we treat the disc robot as a point equipped with position and touch sensors, moving in a planar c-space amidst unknown c-obstacles. A c-space *disconnection path* starts at a configuration S and con-

tains a c-obstacle boundary that separates S from T . The first lemma establishes that *CBUG* terminates once a disconnection path appears in its current bounding ellipse.

Lemma 4.2 *CBUG terminates with a conclusion that T is unreachable in the first bounding ellipse whose interior contains a disconnection path starting from S .*

The lemma is based on the following argument. The sub-algorithm *BUG1* is known to be complete both in terms of its connection and disconnection paths [17]. Once a disconnection path which starts at S lies in the interior of the current bounding ellipse, *BUG1* finds this path and terminates *CBUG* with a conclusion that T is unreachable from S .

The next lemma discusses the possible cases of target unreachability. Let \mathcal{U} be the connected component of the free c-space containing the start S . In general, \mathcal{U} is bounded from the outside by an outer c-obstacle, and is punctured from the inside by internal c-obstacles. The first case of target unreachability occurs when T lies beyond the outer boundary. In this case any disconnection path must circumnavigate the outer boundary of \mathcal{U} . The second case occurs when T lies inside a puncture of \mathcal{U} . In this case any disconnection path must circumnavigate the internal c-obstacle boundary. The two cases are discussed in the following lemma.

Lemma 4.3 *Let α be the shortest off-line disconnection path from S , of length λ_{opt} . If T lies beyond the outer boundary of \mathcal{U} , α lies in a disc with center at S and radius $\lambda_{opt}/2$. If T lies inside a puncture of \mathcal{U} , α lies in an ellipse with focal points S and T and major axis of length λ_{opt} .*

Proof: First consider the case where T lies beyond the outer boundary of \mathcal{U} , denoted β . We may assume that β is a simple closed loop. Since T lies beyond β , any disconnection path from S must contain the entire loop β . Since the length of α is λ_{opt} , the length of β is at most λ_{opt} . Consider now the collection of all loops of length at most λ_{opt} surrounding S . Clearly, a disc with center at S and radius $\lambda_{opt}/2$ contains all such loops. In particular it contains the loop β , and consequently it contains the path α .

Next consider the case where T lies in a puncture of \mathcal{U} . Let γ denote the puncture's boundary, which we assume is a topological loop. The entire γ is part of any disconnection path starting at S . In particular, the shortest disconnection path α starts at S , contains the loop γ , and has total length λ_{opt} . It follows that all points x along α satisfy $\|x - S\| + \|x - T\| \leq \lambda_{opt}$. The latter inequality defines a region bounded by an ellipse with focal points S and T and major axis of length λ_{opt} . \square

The next lemma gives an upper bound on the first bounding ellipse of *CBUG* which contains a disconnection path starting from S .

Lemma 4.4 *The CBUG algorithm terminates with a conclusion that T is unreachable in a bounding ellipse whose area $A(n)$ is bounded by $A(n) <$*

$\frac{\pi}{2}(\lambda_{opt} + \|S-T\|)(\lambda_{opt}^2 + 2\lambda_{opt}\|S-T\|)^{1/2}$, where λ_{opt} is length of the shortest off-line disconnection path which starts at S .

Proof: Based on Lemma 4.2, *CBUG* terminates with a conclusion of target unreachability once its bounding ellipse contains a disconnection path starting at S . It can be verified that the disc and ellipse of Lemma 4.3 are contained in a larger ellipse with focal points S and T and major axis of length $2a = \lambda_{opt} + \|S-T\|$. The latter ellipse contains the shortest off-line disconnection path from S . It can be verified that the ellipse's minor axis has length $2b = (\lambda_{opt}^2 + 2\lambda_{opt}\|S-T\|)^{1/2}$. The ellipse's area is given by πab . Since *CBUG* doubles the area of its bounding ellipse in each iteration, $A(n) \leq 2\pi ab$. Substituting for a and b in the latter inequality gives the bound on $A(n)$. \square

Next we convert the bound on $A(n)$ to a bound on path length. The conversion is based on a key geometric fact for which we need the notion of traceable obstacles. Let \mathcal{CB}_i be the c-space obstacle induced by an obstacle \mathcal{B}_i for a disc robot of size D . The *traceable obstacle* induced by \mathcal{B}_i , denote \mathbf{B}_i , is the physical obstacle obtained by filling any internal holes in \mathcal{CB}_i and then shrinking \mathcal{CB}_i inward by a distance of $D/2$. An important property of \mathbf{B}_i is that the area swept by the disc robot during tracing of its boundary is precisely the area swept by the robot while tracing the boundary of the original obstacle \mathcal{B}_i .

Lemma 4.5 ([12]) *Let a planar environment contain disjoint traceable obstacles $\mathcal{B}_1, \dots, \mathcal{B}_k$. Let a size D disc robot trace the i^{th} obstacle boundary, and let κ_i be the area swept by the robot during this tracing. Let \mathcal{C} be any simple closed curve surrounding the k regions swept by the robot. Then $\sum_{i=1}^k \kappa_i \leq 4A(\mathcal{C})$, where $A(\mathcal{C})$ is the area of the obstacle-free points enclosed by \mathcal{C} .*

The following proposition establishes a quadratic upper bound on the disconnection paths generated by *CBUG*.

Proposition 4.6 *If T is not reachable from S , the basic *CBUG* algorithm concludes target unreachability along a path whose length λ satisfies the quadratic upper bound,*

$$\lambda \leq \frac{6\pi}{D}(\lambda_{opt} + \|S-T\|)^2 + \|S-T\| + \frac{6A_0}{D}, \quad (3)$$

where λ_{opt} is length of the shortest off-line disconnection path from S , D is the disc-robot size, and A_0 is area of the initial ellipse.

Note that the three summands have units of length, so the upper bound is scalable.

Proof: At the i^{th} stage of *CBUG* the robot executes the sub-algorithm *BUG1* in an ellipse with focal points S and T and area $A(i)$. The regions swept by the robot during circumnavigation of obstacles in this ellipse (including the obstacle formed by the ellipse) are surrounded by the ellipse's

boundary. Identifying the latter boundary with the curve \mathcal{C} of Lemma 4.5, the total length of the robot's path during circumnavigation of the obstacles is at most $4A(i)/D$. Recall now that under *BUG1* the robot circumnavigates the boundary of each obstacle at most 1.5 times. Hence the total length of the robot's path during boundary following is at most $6A(i)/D$. Under *BUG1* motion between obstacles is always directly to the target. The total length of these motion segments equals to the net decrease of the robot's distance from T , which is $\|S_i - T\| - \|S_{i+1} - T\|$. Adding the two terms gives the path length bound: $\lambda_i \leq 6A(i)/D + (\|S_i - T\| - \|S_{i+1} - T\|)$.

Suppose that *CBUG* finds a disconnection path at the n^{th} stage, such that $n > 1$. Since the area of the ellipses doubles in each step, $\sum_{i=1}^n A(i) = A_0 + 2A_0 + \dots + 2^{n-1}A_0$, where A_0 is the area of the initial ellipse. Since $A(n) = 2^{n-1}A_0$, we see that $\sum_{i=1}^n A(i) = \sum_{i=1}^{n-1} A(i) + A(n) < 2A(n)$. According to Lemma 4.4, the area of the n^{th} ellipse satisfies the inequality $A(n) < \frac{\pi}{2}(\lambda_{opt} + \|S - T\|)(\lambda_{opt}^2 + 2\lambda_{opt}\|S - T\|)^{1/2} \leq \frac{\pi}{2}(\lambda_{opt} + \|S - T\|)^2$. Hence the total length of the path traveled by the robot is bounded by $\lambda = \sum_{i=1}^n \lambda_i \leq \frac{6}{D} \sum_{i=1}^n A(i) + \sum_{i=1}^n (\|S_i - T\| - \|S_{i+1} - T\|) < \frac{6\pi}{D}(\lambda_{opt} + \|S - T\|)^2 + \|S - T\|$, where we substituted $S_1 = S$ and $S_{n+1} = T$. Finally, the term $6A_0/D$ bounds the path traveled by the robot in the case where the initial ellipse already contains a disconnection path from S . \square

The final result is a universal lower bound on disconnection competitiveness.

Proposition 4.7 *Any navigation algorithm in unknown planar environments generates in worst case a disconnection path whose length satisfies the quadratic lower bound*

$$\lambda \geq \frac{4\pi}{3(1+2\pi)^2 D} (1-\epsilon)\lambda_{opt}^2, \quad (4)$$

where λ_{opt} is length of the shortest off-line disconnection path from S , D is the disc-robot size, and $\epsilon > 0$ is an arbitrary small constant.

Proof sketch: We use the environment which is used to prove the universal lower bound on connection competitiveness [13]. This environment consists of radial corridors emanating from S and having length r . The radial corridors are surrounded by a circular corridor such that only one radial corridor enters the circular corridor. The target is placed in the circular corridor. The shortest off-line path from S to T satisfies $l_{opt} \leq (1 + \pi)r$, where πr is due to worst case motion in the circular corridor to a target located opposite the entry. Not knowing which radial corridor leads to the circular corridor, any on-line algorithm would guide the robot in worst case through all radial corridors before finding the entry to the circular corridor. It is shown in Ref. [13] that any on-line path from S to T in this environment has length l satisfying $l \geq c(1-\epsilon)r^2$, where $c = 4\pi/3D$ and ϵ is an arbitrary small constant.

We now render the target inaccessible in two ways. First we place T outside the circular corridor so that it becomes inaccessible from S . In this case the shortest off-line disconnection path requires radial motion from S to the

circular corridor, then a complete circumnavigation of the circular corridor. In the second case we place T in the circular corridor and surround it by walls located δ apart. If the circular corridor is sufficiently wide, T lies in a puncture of the region accessible from S . In this case the shortest off-line disconnection path requires radial motion from S to the circular corridor, then motion in the circular corridor until the walls surrounding T are met. Combining the two cases, $\lambda_{opt} \leq (1+2\pi)r$ in this environment. Any on-line algorithm must explore in worst case all radial corridors before finding the entry to the circular corridor. Then it must circumnavigate in worst case the entire circular corridor. Hence $\lambda \geq c(1-\epsilon)r^2 + 2\pi r$. Since $r \geq \lambda_{opt}/(1+2\pi)$, we obtain the lower bound $\lambda \geq c_1(1-\epsilon)\lambda_{opt}^2 + c_2\lambda_{opt}$, where $c_1 = 4\pi/3(1+2\pi)^2D$ and $c_2 = 2\pi/(1+2\pi)$. The latter inequality implies that $\lambda \geq c_1(1-\epsilon)\lambda_{opt}^2$. \square

The universal lower bounds (2) and (4) imply that *CBUG* has the lowest possible connection as well as disconnection competitiveness.

5 Significance of Double Competitiveness

This section discusses some useful properties of double competitiveness. First and foremost, double competitiveness ensures that an on-line navigation algorithm would not stray away from the optimal path due to misleading sensory clues. Consider for instance the office floor environment shown in Figure 3. The paths generated by *ALG1* (reviewed below) may stray along the outer walls arbitrarily far from S and T . When *CBUG* runs *ALG1* as a sub-algorithm, the search is confined to the ellipses which are depicted in the figure.

Double competitiveness additionally provides some amount of path stability. A navigation algorithm possesses *path stability* when its path varies continuously with the position of S and T [11]. A weaker notion of path stability is as follows. A navigation algorithm possesses *path length stability* when its path length varies continuously with S and T . This means that small changes in S or T yield small changes in path length and hence travel time from S to T . Classical on-line navigation algorithms such as *BUG1* and *ALG1* respond to small changes in S or T with possibly unbounded path-length changes (Figure 3). However, the competitive quantities l_{opt} and λ_{opt} are approximately constant with respect to small changes of S and T . Hence competitive bounds in terms of l_{opt} and λ_{opt} automatically provide path-length bounds in response to small changes of S and T . Under *CBUG* the length of all connection paths from S to T vary in the bounded interval $[l_{opt}, c_1l_{opt}^2 + c_2]$, where c_1 and c_2 are constants. Similarly, the length of all disconnection paths from S vary in the bounded interval $[\lambda_{opt}, c'_1\lambda_{opt}^2 + c'_2]$, where c'_1 and c'_2 are constants.

Next we describe simulations comparing the non-competitive algorithms *BUG1* and *ALG1* with *CBUG*. The algorithm *ALG1* relies on the *M-line* passing through S and T , and works as follows [20]. The robot moves from S along the *M-line* towards T until it hits an obstacle. Then it circumnavigates the

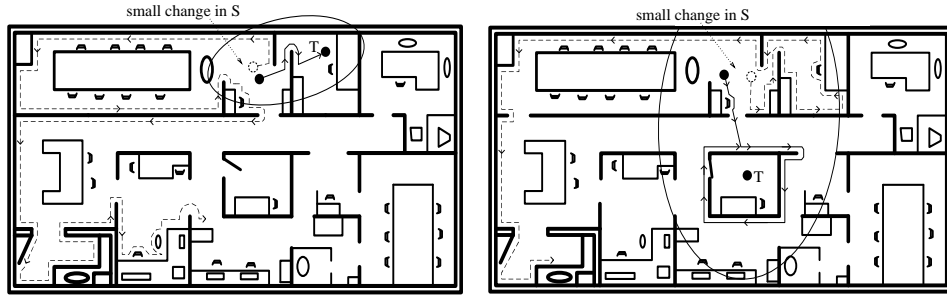


Fig. 3. The paths generated by *ALG1* show significant path-length jumps in response to small changes in S . The ellipses show the total search area of *CBUG*.

obstacle in a clockwise direction. Whenever the robot reaches an intersection point of the current obstacle boundary with the M-line, denoted p , it leaves the obstacle if two conditions are met. The point p must be closer to T than all previous hit and leave points, and the direction from p to T must point away from the current obstacle. Once the robot leaves an obstacle it resumes motion along the M-line until the next obstacle is encountered or the target is found. If p is not a valid leave point but is a previously defined hit or leave point, the robot *reverses* its boundary tracing direction at p . However, the robot is not allowed any further direction reversals along the current boundary segment. When the robot completes a loop around the current obstacle boundary without finding a suitable leave point, it halts with a conclusion of target unreachability. Several paths of *ALG1* are depicted in Figure 3.

In general, the double competitiveness of *CBUG* comes with an overhead incurred by its search ellipses. In order to study the effect of this overhead on average performance, we compared *BUG1* and *ALG1* to *CBUG* implementing *BUG1* and *ALG1* as sub-algorithms. We tested the algorithms in the office floor environment depicted in Figure 3. We placed S and T in three distance ranges: $\|S - T\| \leq 10D$, $10D < \|S - T\| \leq 50D$, and $\|S - T\| > 50D$, where D is the robot size. Each distance range included 30 runs with S and T varying within the prescribed range. Each 30 runs were subdivided into 10-run batches corresponding to three furniture occupancy levels of the office floor.

The results listed in Table 1 give the average ratio l/l_{opt} , where l is length of the path generated by the algorithm and l_{opt} is length of the shortest off-line path from S to T . In the highest distance range S and T were roughly at opposite corners of the office floor. In this case *CBUG* is inferior to *BUG1* and *ALG1*. In this case l_{opt} is not much shorter than the path along the outer walls persistently traced by *BUG1*. The superior performance of *ALG1* is due to its boundary tracing rule, which typically limits its wall tracing to a single room. However, the advantage of *BUG1* and *ALG1* diminishes as T moves closer to S . In the lowest distance range S and T were typically in neighboring rooms. In this case *BUG1* still follows the entire outer walls. Similarly, when T is

Distance Range	<i>BUG1</i>	<i>ALG1</i>	<i>CBUG</i> with <i>BUG1</i>	<i>CBUG</i> with <i>ALG1</i>
$\ S - T\ > 50D$	7.1	3.0	9.0	10.9
$10D < \ S - T\ \leq 50D$	11.5	4.6	7.2	12.0
$\ S - T\ \leq 10D$	28.8	14.1	3.5	7.3

Table 1. Summary of simulation results comparing *CBUG* to *BUG1* and *ALG1*.

located just on the other side of a wall, *ALG1* guides the robot along the entire outer walls. In contrast, *CBUG* recognizes when both sub-algorithms stray away from *S* and *T*. It cuts short their wall following with the bounding ellipses, thus ensuring paths whose average length is twice shorter than the paths of *ALG1* and eight times shorter than the ones generated by *BUG1*. Simulations of cases where *T* is inaccessible from *S* are under preparation and will be discussed in an extended version of this paper.

6 Conclusion

This paper introduced a notion of disconnection competitiveness complementary to the traditional notion of connection competitiveness. Connection competitiveness concerns cases where *T* is reachable from *S*, while disconnection competitiveness concerns cases where *T* cannot be reached from *S*. We described a tactile-sensor based navigation algorithm for a disc-shaped robot, called *CBUG*. The algorithm achieves connection as well as disconnection competitiveness by limiting its search to a series of expanding ellipses. The connection competitiveness of *CBUG* is quadratic in l_{opt} , which matches up to constants the universal lower bound over all on-line navigation algorithms. The disconnection competitiveness of *CBUG* is quadratic in λ_{opt} , where λ_{opt} is the shortest off-line disconnection path starting from the same *S*. The universal lower bound on disconnection competitiveness over all on-line navigation algorithms is also quadratic in λ_{opt} . Hence up to constants *CBUG* has tight connection as well as disconnection competitiveness. However, competitiveness concerns worst case behavior, and does not necessarily indicate efficient average behavior. Simulations reveal that in practice *CBUG* may incur significant overhead in certain situations.

Two over-simplifications of the navigation problem discussed in this paper are as follows. First, mobile robots are usually not disc-shaped but rather rigid bodies having three degrees of freedom. Our preliminary work on three degrees-of-freedom mobile robots indicates that on-line navigation of simple shapes can be achieved with *cubic* competitiveness. Second, *CBUG* assumes tactile sensors. More sophisticated sensors such as vision and laser sensors do not have a significant advantage over tactile sensors in highly congested environments. However, practical environments tend to be reasonably sparse, and an adaptation of *CBUG* to such sensors is an important open problem. Last, the constants in the quadratic upper bounds on *CBUG* differ from their

corresponding quadratic universal lower bounds by a factor of about 100. The closing of this gap is a major challenge that can yield new algorithms with an improved average performance.

References

1. R. Baeza-Yates, J. Culderon, and G. Rawline. Searching in the plane. *J. of Information and Computation*, 106:234–252, 1993.
2. J. Basch, L. J. Guibas, D. Hsu, and A. T. Nguyen. Disconnection proofs for motion planning. In *IEEE Int. Conf. on Robotics and Automation*, pages 1765–1772, 2001.
3. R. Bellman. Problem 63-9. *SIAM Review*, 5(2), 1963.
4. P. Berman, A. Blum, A. Fiat, H. Karloff, A. Rosen, and M. Saks. Randomized robot navigation algorithms. In *SODA*, pages 75–84, 1996.
5. A. Blum, P. Raghavan, and B. Schieber. Navigating in unfamiliar terrain. In *STOC*, pages 494–504, 1991.
6. H. Choset and J. W. Burdick. Sensor based planning, part ii: Incremental construction of the generalized voronoi graph. In *IEEE Int. Conference on Robotics and Automation*, pages 1643–1649, 1995.
7. H. Choset, K. M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion*. MIT Press, Cambridge MA, 2005.
8. A. Datta and C. Icking. Competitive searching in a generalized street. In *10th ACM Symp. on Computational Geometry*, pages 175–182, 1994.
9. A. Datta and S. Soundaralakshmi. Motion planning in an unknown polygonal environment with bounded performance guarantee. In *IEEE Int. Conf. on Robotics and Automation*, pages 1032–1037, 1999.
10. E. Bar Eli, P. Berman, A. Fiat, and P. Yan. Online navigation in a room. *J. of Algorithms*, 17(3):319–341, 1996.
11. M. Farber. Instabilities of robot motion. *Topology and its Applications*, 140:245–266, 2004.
12. Y. Gabriely and E. Rimon. Competitive complexity of mobile robot on-line motion planning problems. In *6th Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 155–170, 2004.
13. Y. Gabriely and E. Rimon. Cbug: A quadratically competitive mobile robot navigation algorithm. In *IEEE Int. Conf. on Robotics and Automation*, pages 954–960, 2005.
14. C. Icking, R. Klein, and E. Langetepe. An optimal competitive strategy for walking in streets. *16th Symp. on Theoretical Aspects of Computer Science*, 110:405–413, 1988.
15. I. Kamon, E. Rimon, and E. Rivlin. Tangentbug: A range-sensor based navigation algorithm. *Int. Journal of Robotics Research*, 17(9):934–953, 1998.
16. S. L. Laubach, J. W. Burdick, and L. Matthies. An autonomous path planner implemented on the rocky7 prototype microrover. In *IEEE Int. Conf. on Robotics and Automation*, pages 292–297, 1998.
17. V. J. Lumelsky and A. Stepanov. Path planning strategies for point automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2:403–430, 1987.

18. H. Noborio and T. Yoshioka. An on-line and deadlock-free path-planning algorithm based on world topology. In *Conf. on Intelligent Robots and Systems, IROS*, pages 1425–1430. IEEE/RSJ, 1993.
19. C. H. Papadimitriou and M. Yannakakis. Shortest paths without a map. *Theoretical Computer Science*, 84:127–150, 1991.
20. A. Sankaranarayanan and M. Vidyasagar. Path planning for moving a point object amidst unknown obstacles in a plane: the universal lower bound on worst case path lengths and a classification of algorithms. In *IEEE Int. Conf. on Robotics and Automation*, pages 1734–1941, 1991.
21. B. Tovar, S. M. Lavalle, and R. Murrieta. Optimal navigation and object finding without geometric maps or localization. In *IEEE Int. Conf. on Robotics and Automation*, pages 464–470, 2003.