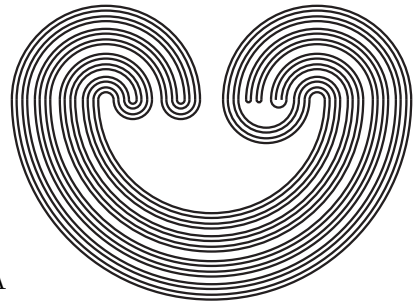
Topology Proceedings



Web: http://topology.auburn.edu/tp/
Mail: Topology Proceedings
Department of Mathematics & Statistics
Auburn University, Alabama 36849, USA
E-mail: topolog@auburn.edu
ISSN: 0146-4124

Topology Proceedings
Vol 17,1992

Invitied Speaker

TOPOLOGICAL APPROACHES TOWARD PROVING LOWER BOUNDS IN COMPUTER SCIENCE

MICHAEL D. HIRSCH

ABSTRACT. Proving lower bounds for the cost of solving problems on a computer is a very difficult problem—not least because it is a global problem: one must minimize cost over all possible algorithms. As with many other global problems, a topological approach has proven fruitful. We give two applications of topology to the problem of proving lower bounds. The first is to the computation of the *topological complexity* of a problem. We define the New Point Problem and compute tight bounds for its topological complexity. The second application is to finding lower bounds on *one way communication complexity* of two processors.

1. INTRODUCTION

When analyzing any problem in computer science, there are three questions that are asked. These are the questions of

(1) Solvability: Is it possible to solve the given problem with a computer?

(2) Upper bounds: What is an upper limit to the number of operations the computer must perform to solve the problem in the worst case?

(3) Lower bounds: What is a lower bound to the number of operations necessary to solve the problem in the worst case?

Solvability is often quite easy to decide. For many problems, especially for simple computational questions such as sorting or matrix multiplication, there are obvious algorithms. These algorithms were know long before there were computers and it is easy to see that a computer can successfully perform them.

Other problems, for example determining whether a system of equations has a solution, are major mathematical questions. Until this question was solved by Tarski[13] it was not at all clear that this was possible to do algorithmically.

Of course, there are still significant areas in which this question is unanswered. Whether artificial intelligence is possible is an article of faith for many workers in the field, but it is still hotly debated in philosophical circles.

Having answered the question of solvability, proving an upper bound is often straightforward as well. One analyzes the known algorithm and proves that it always terminates after a certain number of operations. (Usually one speaks of "cost" instead of "number of operations". The usual assumption is that each operation costs one unit. A more realistic model might charge twice as much for multiplication as for addition since multiplying usually takes longer than adding. One can then define "the cost due to multiplying", "the cost due to adding", etc. in the obvious way.)

Most upper bounds are of this form. It is rare, though not unheard of, to prove an upper bound for an unknown algorithm. There are occasional non-constructive proofs of the existence of efficient algorithms, but they are rare.

The third question, that of finding lower bounds for algorithms, is very hard. Somehow one must prove that all algorithms, including those you don't know, must perform a certain number of operations to solve the given problem. This, naturally, is significantly harder than finding and analyzing an algorithm.

In general, computer scientists believe that the first two problems have been pretty well solved for most computational problems. They believe that the known algorithms are close

to optimal. Ideally, the upper bound should match the lower bound. Then we would know our algorithm is as fast as it could ever be and we could look for new problems. There are some problems with matching upper and lower bounds, but this is not the general case. Ben-Or [3], building on Steele and Yao's work [12], gives a general technique for proving lower bounds and provides many examples of both optimal and non-optimal lower bounds.

Problems are are typically classified into those which are solvable in polynomial cost, and those which aren't. There is a large and important (both theoretically and practically) class of problems, the *NP-complete* problems, which are generally believed to be exponential cost. Garey and Johnson [5] is a good book defining NP-completeness and provides a list of over 300 such problems.

Certainly every known algorithm for an NP-complete problem has worst case exponential cost. However, the best known lower bounds tend to be low order polynomials. For instance, all algorithms for the Knapsack Problem (given a list of n numbers, is there a subset which adds up to another given number?) take a worst case cost which is exponential in n, but the best lower bound is only $n^2/2$ [3].

The questions of solvability and upper bounds are *local* in nature–they are questions about particular known objects. The problem of finding lower bounds is *global* in nature–it is about all objects of a given type. It seems natural to try to harness topological techniques to this question. Topology has a long history of successful application to global questions, from the Bridges of Königsberg to modern Global Analysis.

The techniques of Ben-Or [3] and Steel and Yao [12] mentioned above do, in fact, hinge upon topological results of Milnor [9] and Thom [14] on the Betti numbers of a solution of polynomial equations, so this is not exactly a new observation. Björner, Lovász and Yao [4] have another approach which gives a lower bound in terms of Euler characteristics.

In this paper we will present two areas in which topology can provide interesting lower bounds. The first is the topological complexity of a problem, which is fundamentally related to the topology of the inputs and outputs of the problem. This was first studied by Smale [11] for root finding, then by Vasiliav [15] and Levine [7]. We will define a new problem, the New Point Problem, and compute its topological complexity for a variety of spaces. These results are written up in more detail in [6].

The second application of topology will be to communication complexity. Here we have two computers trying to compute some function of their inputs and we wish to know how many pieces of data they must communicate to each other. This has been widely studied in the discrete model of computation and less widely in the continuous model. In the continuous model a topological approach provides an good intuitive framework. We will prove Abelson's theorem [2], and also prove the version given by Luo and Tsitsiklis [8], in a manner which is particularly simple and intuitive.

2. Topological Complexity

In this section we discuss the application of topology to the problem of finding the topological complexity of a problem. The term "topological complexity" was invented by Smale in the study of root finding [11].

Topological complexity is the number of branching statements to solve a problem. A typical algorithm is full of statements like "if $x > 0$ then do this, otherwise to that." To make this rigorous we must define our model of computation.

2.1. The Model Of Computation.

Definition 2.1. *An algorithm is a binary tree where the input is n real numbers and at each node a rational function of the input is calculated, then compared to zero. The algorithm goes down one branch or the other depending on the outcome of*

the comparison. In a leaf a rational function of the inputs is computed.

Figure 2.1 is an example of an algorithm for computing whether $|x| > 1$. Here YES and NO are thought of as constant functions with some coding for YES and NO, say YES = 1, NO = 0.

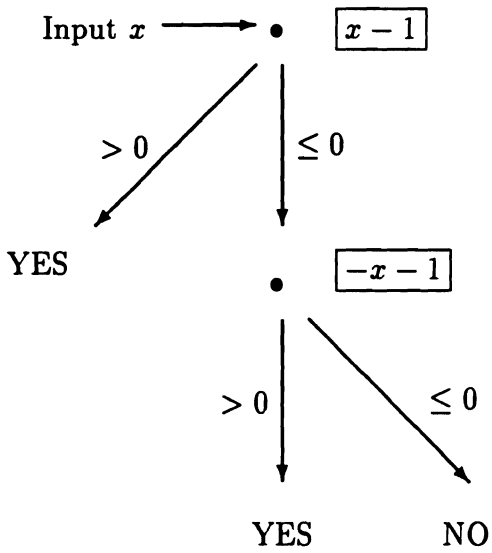


FIGURE 1. Testing $|x| > 1$ with 2 branches, no multiplications

Actually, the results we will get can be obtained with the assumption that all functions are continuous functions. In this model at each node an arbitrary continuous function of the inputs is computed and compared to zero.

Note that this model of computation is not well suited for calculating total cost of a computation as many computations are done at each node. It is, however, well suited for the study of topological complexity.

Definition 2.2. *The* topological complexity of an algorithm *T is the number of branch nodes in T. Note that this is the number of leaves - 1.*

The topological complexity of a problem *is the minimum, over all algorithms which solve the problem, of the topological complexity of each algorithm.*

Many problems have trivial topological complexity. For instance, finding the determinant of an $n \times n$ matrix takes no branching though this method might have unnecessarily high cost. It might be more efficient to do some row and column reductions so as to introduce zeros and simplify the resulting formula, thus lowering the cost of the computation, but raising the topological complexity of the algorithm.

Because many computational questions can be answered without branching, topological complexity is a more interesting concept for search problems, i.e., problems for which there are many possible answers.

Example 2.3 Sorting is a simple example of topological complexity in a search problem. Assuming that each leaf computes a rational function of the inputs which sorts the inputs, each rational function is just a permutation of the inputs. Thus there must be as many leaves as there are possible orders for the inputs, so the topological complexity is $n! - 1$. To have that many branching nodes the algorithm must have depth $\log_2(n!) \cong n \log_2 n$. As there are algorithms of depth $\mathcal{O}(n \log_2 n)$ we have upper and lower bounds of the same order.

2.1.1 Tradeoffs between Computing and Branching

Ben-Or's techniques for proving lower bounds [3] give lower bounds for the number of multiplications, divisions and branching all together. Interestingly, there is often a tradeoff between multiplication/division and branching. For example, consider Figure 2.1. It is an algorithm for testing whether

$|x| > 1$. It has two branch nodes and no multiplications. Figure 2.2 is another algorithm for the same test, this time with only one branch node but with one multiplication.

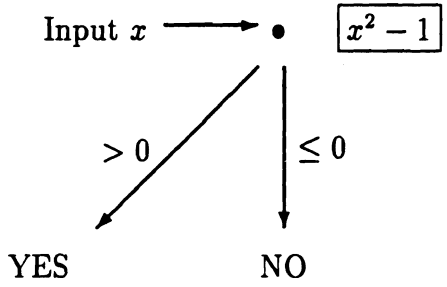


FIGURE 2. Testing $|x| > 1$ with 1 branch, 1 multiplication

We would like to know to what extent branching and multiplying can be traded off. Clearly you can't compute x^{n^2} without some multiplications, nor can you sort without branching, but there is no systematic way of determining how many of each are necessary. This work is an start in developing such systematics.

2.1.2 Leaf Spaces

We can now rephrase topological complexity into topological terms.

Definition 2.4. *Let $\mathcal{I}$ be the set of all possible inputs to a given algorithm. For a given leaf ℓ, the* leaf space *of ℓ, V_ℓ, is the set of points of $\mathcal{I}$ for which the algorithm ends at ℓ.*

Since every point in the input space ends up at some leaf, an algorithm yields a decomposition of $\mathcal{I}$ into leaf spaces V_ℓ, and on each V_ℓ there is a function which computes a continuous solution to the problem. Thus, the answer to the question "What is the topological complexity of a problem?" can be bounded below by the answering the the question "What is the minimal number of sets V_ℓ that $\mathcal{I}$ can be decomposed into,

so that there is a continuous solution ϕ_ℓ on each V_ℓ?" We have proven

Proposition 2.5. *The topological complexity of a problem* $\geq$ *the number of sets* V_ℓ *needed in a partition of the inputs so that there are continuous functions solving the problem on each* $V_\ell - 1$.

3. The New Point Problem

The New Point Problem (NPP) is a toy problem designed specifically to have unavoidable branching without much computation. In its most applicable form it is known as the *Burger King problem.*

Example 3.1 As owner of all the Burger Kings on Manhattan island, you want to open a new Burger King, but there are some constraints. In order not to compete with yourself, all your restaurants are at least 5 blocks apart. You need to find a place to put your new restaurant at least 5 blocks away from all your other. Leaving aside questions of optimal location, and assuming you haven't filled all the available space, can you find a place for the new Burger King? What is the topological complexity of doing so?

The topological complexity of this problem is in Corollary 3.14, except instead of Manhattan we will use the closed interval. The Corollary will follow from the analysis of a simpler question.

Definition 3.2. *The* New Point Problem, *or NPP: Given a list of points* $x_1, x_2, \ldots, x_n$ *in a topological space* X, *find a new point* $y \in X$ *not in the list.*

Remark 3.3 Here we assume that X is somehow modelable on a computer in such a way that we can cheaply compute whether two points $x, y \in X$ are the same, i.e., for unit "charge" against

the topological complexity of the algorithm. Examples of such spaces are R, $[0,1]$, $(0,1)$, and S^n.

Remark 3.4 It is important that we use a *list* of points and not, for instance, a *set*. In the first place, computers cannot handle unordered collections of elements; all inputs have a first, second, etc., even if there is no meaning attached to the order. Secondly, as we show in Example 3.8 it is important for these results that the elements can be repeated—in fact all the inputs can be the same number.

The topology of the space X has a significant effect on the topological complexity of NPP on X.

Example 3.5 On S^1 when $n = 1$, $x \mapsto -x$ gives a new point. Thus, the topological complexity of NPP on S^1 with $n = 1$ is 0.

Example 3.6 Every continuous map $[0,1] \to [0,1]$ has a fixed point, so no such formula works for $[0,1]$ with $n = 1$. Thus, the topological complexity of NPP on $[0,1]$ when $n = 1$ is 1.

Example 3.7 On R, $\sum_{i=1}^{n} x_i^2 + n$ is always a new point, so the topological complexity is 0. The open interval $(0,1)$ is homeomorphic to R so the topological complexity is 0 on $(0,1)$, too.

Example 3.8 Given n *distinct* points in $[0,1]$ the topological complexity of finding a new point is 0. Let $\{x_i\}$ be the points. Define

$$\epsilon = \prod_{i=2}^{n} x_1 - x_i.$$

Then ϵ is smaller than the distance between x_1 and any other point, so $y = x_1 + \epsilon^2(x_2 - x_1)$ is a new point. Hence, topological complexity is 0 for 2 or more distinct points in $[0,1]$.

3.1. Upper Bounds. There is an trivial upper bound of n for any X modelable on a computer. We make it explicit in the case of $X = [0,1]$.

Theorem 3.9. *The topological complexity of NPP on* $[0,1]$ *is* $\leq n$.

Proof: To prove this we need to present an algorithm which when given $x_1, x_2, \ldots, x_n \in [0,1]$ will find a new point and we can guarantee it has only n branch nodes. Consider the points $\{0, \frac{1}{n}, \frac{2}{n}, \ldots, \frac{n}{n}\}$. One at a time, our algorithm will test the elements of this set against $\{x_1, x_2, \ldots, x_n\}$. Testing can be done in one branch step by testing whether

$$\prod_{i=1}^{n}(x_i - \frac{k}{n}) \stackrel{?}{=} 0.$$

If no, the $\frac{k}{n}$ is a new point. Otherwise $\frac{k}{n}$ is in the list.

After testing the first n tests, assuming we haven't found a new point yet, we are guaranteed the final test point is not in the set. In this case the algorithm outputs the final point without a test. So the algorithm has exactly n branch nodes. □

This argument will obviously work for any metric space with a metric which is computable without branching. Sometimes we can do a bit better.

Theorem 3.10. *NPP on* S^1 *has* $TC \leq n-1$.

Proof: Again, we present an algorithm. Let the inputs be $x_1, x_2, \ldots, x_n$ and let R_θ be rotation by θ.

The algorithm is exactly as in the interval case, but with test points

$$\{R_{2\pi/(n+1)}(x_1), R_{4\pi/(n+1)}(x_1), \ldots, R_{2n\pi/(n+1)}(x_1)\}.$$

We are guaranteed that none of these points is x_1, so we have effectively eliminated it from the problem. The algorithm does the same type of analysis, but in this case the final point is guaranteed to be a new point after only $n-1$ comparisons. □

3.2. Lower Bounds. In this section we will show that the upper bounds of the previous section are tight. We will start with $[0,1]$.

Theorem 3.11. *The topological complexity of NPP on* $[0,1]$ *is* n.

Proof: Suppose not. Then

$$[0,1]^n = \bigcup_{\ell=1}^{n} V_\ell$$

and we have continuous functions $f_\ell : V_\ell \to [0,1]$ with

$$f_\ell(x_1, \ldots, x_n) \neq x_j \text{ for all } \ell, j.$$

Note that if there are fewer than $n-1$ branch nodes some of the V_ℓ will be empty.

Extend each f_ℓ to an open set U_ℓ preserving

$$f_\ell(x_1, \ldots, x_n) \neq x_j \text{ for all } \ell, j.$$

Abuse notation and call the resulting functions f_ℓ. This is trivial when the f_ℓ are rational functions. When we only assume continuity of the f_ℓ there are some technical details which are worked out in [6].

Let $C_\ell \subset U_\ell$ be a closed cover of $[0,1]^n$. Restricting f_ℓ to these sets we have

$$[0,1]^n = \bigcup_{\ell=1}^{n} C_\ell$$

and we have continuous functions $f_\ell : V_\ell \to [0,1]$ with

$$f_\ell(x_1, \ldots, x_n) \neq x_j \text{ for all } \ell, j.$$

By the Tietze Extension Theorem extend the f_ℓ to $[0,1]^n$, again calling the resulting functions f_ℓ. Let $F = (f_1, f_2, \ldots, f_n) : [0,1]^n \to [0,1]^n$.

Then for all $x \in [0,1]^n, F(x) \neq x$ because $x \in C_\ell$ and

$$f_\ell(x_1, \ldots, x_n) \neq x_\ell \text{ for all } x \in C_\ell,$$

contradicting the Brouwer Fixed Point Theorem. □

Remark 3.12 We have used the topology of $[0,1]$ twice in the proof of Theorem 3.11, once when we applied the Tietze Extension Theorem, and once to get the contradiction with the Brouwer Fixed Point Theorem. It is clear that the same proof works for any finite dimensional closed ball. We will give some consequences of Theorem 3.11 to other versions of NPP, including NPP on S^1, but it is not so clear how to extend techniques to other spaces which don't satisfy these properties.

Corollary 3.13. *NPP with inputs in sorted order has topological complexity* $= n$.

Proof: Sorting the inputs is a continuous function, thus if there were an algorithm which solves the problem with sorted inputs in fewer than n branch nodes, there would be an algorithm (of the generalized type, i.e, with continuous functions computed at each node) for the unsorted NPP. First it would sort the input (no branching required in this model), then it would solve the sorted NPP with $\leq n-1$ branch node contradicting Theorem 3.11. □

Corollary 3.14. *The Burger King problem on* $[0,1]$ *has topological complexity* $= n$.

Proof: In this problem we have $x_1, x_2, \ldots, x_n \in [0,1]$ such that $|x_i - x_j| > \delta$ if $i \neq j$ and we want y which also satisfies $|y - x_i| > \delta$ for all i. We will reduce the sorted NPP on $[0, 1-2n\delta]$ to this problem and apply Corollary 3.13.

Assume we are given $x_1 \leq x_2 \leq \cdots \leq x_n \in [0, 1-2n\delta]$. We can map them to the inputs of the Burger King problem on $[0,1]$ by the continuous map $f(x_1, x_2, \ldots, x_n) = (x_1 + \delta, x_2 + 3\delta, \ldots, x_n + (2n-1)\delta)$.

Now use an algorithm for the Burger King problem to find a solution $y \in [0, x_1) \cup (x_1 + 2\delta, x_2 - 2\delta) \cup \cdots \cup (x_n + 2n\delta, 1]$ where some of these intervals may be empty. This is homeomorphic to to space of solutions to the original problem, $[0, x_1) \cup (x_1, x_2) \cup \cdots \cup (x_n, 1 - 2n\delta]$, by a map which is easily

seen to vary continuously (but not rationally) as a function of $(x_1, x_2, \ldots, x_n)$.

Thus the topological complexity of the Burger King Problem is at least as great as that of the standard NPP. □

Corollary 3.15. *NPP in the circle has topological complexity* $= n - 1$.

Proof: Represent the circle as $[0, 2\pi]/0 \sim 2\pi$. Given any $n - 1$ points $x_1, x_2, \ldots, x_{n-1} \in [0, 2\pi]$ map them to the n points $[0], [x_1], [x_2], \ldots, [x_{n-1}]$ in S^1 where $[\cdot]$ represents the element under the equivalence relation $\sim$.

A solution to NPP on S^1 with these inputs corresponds to exactly one point in $[0, 1] - \{0, x_1, x_2, \ldots, x_{n-1}\}$ and the correspondence map varies continuously in the original inputs. Thus, we can't solve the NPP in S^1 with less than $n-1$ branch nodes. □

3.3. Abstract Problems. If one wishes to solve NPP on spaces more complicated than $[0, 1]$ or S^1, say on the set of k-planes in $\boldsymbol{R}^n$ or a Lie group, it would seem that more powerful techniques are needed. We will need more powerful techniques and this requires a more topological setting.

In this section we will define a general problem and indicate how Algebraic topology can be applied to solve the problem.

Definition 3.16. *A* problem *is a map* $\pi : \mathcal{O} \to \mathcal{I}$ *from output to input which you wish to invert.*

Example 3.17 In [11], Smale was the first to define a problem this way. He was interested in the topological complexity of finding roots. In this context, the problem of finding the roots of a monic, degree d polynomial over C is the problem of finding a section of the Veita map

$$\begin{array}{c} Roots(d) = C^d \\ \downarrow \pi \\ Poly(d) = C^d \end{array}$$

To make this definition work with all problems, note that it is not necessary to forget the original inputs when you know the answer. Thus the output space $\mathcal{O}$ can be intuitively written $\mathcal{O} = \{(inputs, solutions)\}$. The map to the input space $\mathcal{I}$ is just projection.

In this context, NPP on some manifold M is the map

$$\begin{array}{c} M \times M \times \cdots \times M \times M - \Delta \\ \downarrow \pi \\ M \times M \times \cdots \times M \end{array}$$

where

$$\Delta = \{(x_1, \ldots x_n, x_{n+1}) \,|\, x_i = x_{n+1}, 1 \leq i \leq n\}.$$

Figure 3.3 is a picture of NPP on the interval for the case $n = 2$. Clearly, there is no section of the map π which doesn't cross the diagonal. We would like to prove lower bounds on the number of sets needed to cover M so that π has a section on each element of the cover.

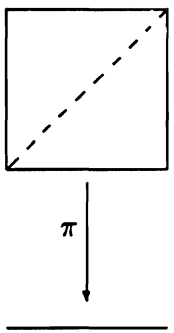


Figure 3.3: NPP on the interval for $n = 2$.

It turns out that Schwarz proved a theorem in 1961 [10], rediscovered by Smale [11], which applies to exactly this sort

of situation. First, we need a definition.

Definition 3.18. *The* cup-length *of a ring is the maximum number of elements of the ring which can be multiplied to get a non-zero number.*

Theorem 3.19. (Schwarz) *Let $f : X \to Y$ be a continuous map, and let k be the number of open sets necessary to cover Y so that f has a section over each open set. Then $K = ker(f^* : H^*(Y) \to H^*(X))$ is a ring under the cup product and $k \geq 1+$ cup-length(K).*

Smale [11] used this theorem to obtain lower bounds on the topological complexity of root finding. Vasiliev [15] improved these results by using deeper properties of the Vieta map, but for NPP Schwartz's Theorem proves an almost tight bound.

Theorem 3.20. *The cup-length of the kernel of*

$$\pi^* : H^*(M \times \cdots \times M) \to H^*(M \times \cdots \times M \times M \backslash \Delta)$$

is $\geq n - 1$.

Corollary 3.21. *NPP on any closed manifold has* $TC \geq n-1$.

The proofs are straight forward and will appear in full in [6].

For many manifolds, e.g. any torus, S^3, or more generally any manifold with a non-zero vector field, the trick from the proof of the upper bound on S^1 (Theorem 3.10) can be used to make the upper bound matches this lower bound. Whether the trivial upper bound of n can be lowered to $n - 1$ for other manifolds is an interesting question. P^2 is a particularly interesting space for this problem because, like the interval, it has the fixed point property: Any continuous map $P^2 \to P^2$ has a fixed point.

4. One Way Communication Complexity

Another area in which a topological viewpoint provides a particularly intuitive and simplifying point of view is in the study of One Way Communication Complexity.

In this we have two processors (or people) P_1 and P_2 each of whom have some information in the form of real numbers, and they are trying to compute some function f of their information. Let $x = (x_1, \ldots, x_m)$ be P_1's information and $y = (y_1, \ldots, y_n)$ be P_2's. Perhaps they each have a vector and they want the inner product, or they each have a matrix and they want the determinant of the sum of the matrices.

P_1 is allowed to communicate with P_2 via "messages," each consisting of a real number which is a rational function of P_1's inputs (or any piecewise smooth function). P_2 is then allowed to compute another rational function of the messages and $y_1, \ldots, y_n$ to arrive at $f(x, y)$.

Let $m_1(x), m_2(x), \ldots, m_k(x)$ represent the messages passed from P_1 to P_2. Then the "communication protocol" looks like Figure 4.4. The idea, of course, is to prove lower bounds on k.

$$
\begin{array}{ccc}
x = (x_1, x_2, \ldots, x_m) & & y = (y_1, y_2, \ldots, y_n) \\
\downarrow & & \downarrow \\
P_1 & & P_2 \\
P_1 & \xrightarrow{m_1(x)} & P_2 \\
P_1 & \xrightarrow{m_2(x)} & P_2 \\
 & \vdots & \\
P_1 & \xrightarrow{m_k(x)} & P_2 \\
 & & \downarrow \\
\multicolumn{3}{c}{\tilde{f}(y, m_1(x), m_2(x), \ldots, m_k(x)) = f(x, y)}
\end{array}
$$

FIGURE 3. Figure 4.4: Computing f(x,y) with two processors.

Communication Complexity has been studied widely in discrete models of computation, but less so in continuous models.

Some of the earliest work in Communication Complexity was Abelson [1] [2] and he studied continuous models. More recently Luo and Tsitsiklis [8] have done more work. Both papers prove versions of the Corollary 4.2 to Theorem 4.1. The proof we present here is considerably simpler than their's.

Theorem 4.1. *With P_1, p_2, f, x, y as defined above, let*

$$\phi : \boldsymbol{R}^m \to Maps(\boldsymbol{R}^n, \boldsymbol{R})$$

be defined by

$$x \mapsto f(x, \cdot).$$

Then the communication complexity of computing f is bounded below by Dim(Image(ϕ)).

Proof: Since $y \in \boldsymbol{R}^n$ is arbitrary, P_2 must know $\phi(x)$ to compute $f(x,y)$. $m_1, m_2, ..., m_k$ specify a point in a k-dimensional space. Since we are assuming rational (or piecewise smooth) computations, by invariance of domain if Dim(Image(ϕ)) $> k$ then P_2 can't know $\phi(x)$ for all values of x. □

Corollary 4.2. *Rank($D\phi$) is a lower bound for the communication complexity.*

Proof: Rank($D\phi$) is a lower bound for Dim(Image(ϕ)). □

Example 4.3 Computing the inner product of two vectors. In this case we have $f(x,y) = \sum_{i=1}^{n} x_i y_i$, and $\phi : \boldsymbol{R}^n \to L(\boldsymbol{R}^n, \boldsymbol{R}) \cong \boldsymbol{R}^n$ is the identity map. Thus, Rank($D\phi$) $= n$ and the communication complexity $\geq n$. Since n is obviously an upper bound as well, the bound is tight.

Example 4.4 Computing the polynomial $f(x,y) = x_1y_1 + x_1y_2 + x_2y_1 + x_2y_2$. Again we have, $\phi : \boldsymbol{R}^2 \to L(\boldsymbol{R}^2, \boldsymbol{R}) \cong \boldsymbol{R}^2$, but this time ϕ is not the identity.

$$D\phi = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}$$

and thus the computational complexity is ≥ 1.

Noting $f(x_1, x_2, y_1, y_2) = (x_1 + x_2)(y_1 + y_2)$ we see that there is a protocol with only one message passed, so again we have tight bounds.

ACKNOWLEDGE

This research was partially supported by grants from the Department of Education and the National Science Foundation.

REFERENCES

[1] H. Abelson, *Lower bounds on information transfer in distributed computations*, In IEEE FOCS, (1978), 151–158.

[2] H. Abelson *Lower bounds on information transfer in distributed computations*, Journal of the ACM, **2** (1980), 384–392.

[3] M. Ben-Or, *Lower bounds for algebraic computation trees*, Proceedings 15th ACM STOC, (1983), 80–86.

[4] A. Björner, L. Lovász, and A. C. C. Yao, *Linear decision trees: volume estimates and topological bounds*, In Proceedings 24th ACM STOC, (1972), 170–177.

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability A Guide to the Theory of NP-Completeness*, W. H. Freeman, 1979.

[6] M. D. Hirsch, *Applications of topology to lower bound estimates in computer science*, In M. W. Hirsch, M. Shub, and J. Marsden, editors, From Topology to Computation Proceedings of the Smale-Fest., Springer-Verlag, 1993, 394-418.

[7] H. Levine, *A lower bound for the topological complexity of poly(d, n)*, to appear in *J. of Complexity.*

[8] Z.-Q. Luo and J. N. Tsitsiklis, *Communication complexity of algebraic computation*, In IEEE FOCS, (1990), 758–765.

[9] J. Milnor, *On the Betti-numbers of real varieties*, Proceedings of the American Mathematical Society, **15** (1964), 275–280.

[10] A. S. Schwarz, *The genus of a fibre bundle*, Proceedings of the Moscow Mathematical Society, **10** (1961), 217–272. 1961.

[11] S. Smale, *On the topology of algorithms, I* Journal of Complexity, **3** (1987), 81–89.

[12] M. Steele and A. Yao, *Lower bounds for algebraic decision trees* Journal of Algorithms, **3** (1982), 1–8.

[14] R. Thom, *Sur l'homologie des variétés algébriques réelles*, In S. S. Cairns, editor, *Differential and Combinatorial Topology*, Princeton University Press, 1965, 255–265.

[15] V. Vasiliev, *Cohomology of the braid group and the complexity of algorithms*, preprint.

Emory University
Atlanta, GA 30322
e-mail: hirsch@mathcs.emory.edu