ABSTRACT

Combinatorial optimization is a diverse area of mathematics. It concerns optimization on feasible regions defined by discrete sets, graphs, hypergraphs, matroids, etc... which all have a large number of applications. They occur in virtually all domains of human activity, since humans always want to do things easier, faster, consume less resources, etc... This thesis concerns three applied problems within combinatorial optimization.

The first paper generalizes previous optimal upper bounds on the minimum Euclidean distance for phase-shift keying (PSK) block codes, that are explicit in the parameters alphabet size, block length and code size. There is a strong connection between high minimum Euclidean distance and good error-correcting capabilities. The bounds are generalized in several respects, such as from codes on symmetric PSK to codes on asymmetric PSK. They are also generalized to other types of noise than Gaussian, allowing more efficient block codes when noise is non-Gaussian. We provide examples of codes on asymmetric PSK that have higher minimum Euclidean distance than any comparable codes on symmetric PSK. Several classes of codes are shown to be optimal among codes on symmetric PSK since their Euclidean distance coincides with the bound.

The second paper considers a parallel computer system with identical computers, where we study optimal performance precaution for one possible computer crash. We anticipate that some computer may crash, and restrict the cost in such a situation. How costly is such a precaution when no crash occurs? We set a restriction that the completion time of a parallel program after a crash is at most a factor $r + 1$ larger than if we use an optimal allocation with $m - 1$ computers. This is an $r$-dependent restriction of the set of allocations of a program. Then the worst-case ratio of the optimal $r$-dependent completion time in the case of no crash and the unrestricted optimal completion time defines a function $f(r,m)$. In the paper we establish upper and lower bounds of the worst-case cost function $f(r,m)$ and characterize worst-case programs.

The third paper considers the problem of Map Matching (MM), i.e. matching time and location measurements of a vehicle to a route in a road network. The paper presents a probabilistic algorithm for MM based on a second order hidden Markov model (HMM), as opposed to first order HMMs which are usually used. This allows a more detailed analysis of the data while preserving algorithmic complexity $O(n)$. Both measurement densities and transition probabilities are determined with respect to Kolmogorov’s third axiom, which in this context implies that the probabilities are additive over a partition of a road segment.
Combinatorial Optimization -
Three Applications

Efraim Laksman
Combinatorial Optimization -  
Three Applications

Efraim Laksman

Doctoral Dissertation in  
Mathematics and its Applications

School of Engineering  
Blekinge Institute of Technology  
SWEDEN
Abstract

Combinatorial optimization is a diverse area of mathematics. It concerns optimization on feasible regions defined by discrete sets, graphs, hypergraphs, matroids, etc. . . which all have a large number of applications. They occur in virtually all domains of human activity, since humans always want to do things easier, faster, consume less resources, etc. . . This thesis concerns three applied problems within combinatorial optimization.

The first paper generalizes previous optimal upper bounds on the minimum Euclidean distance for phase-shift keying (PSK) block codes, that are explicit in the parameters alphabet size, block length and code size. There is a strong connection between high minimum Euclidean distance and good error-correcting capabilities. The bounds are generalized in several respects, such as from codes on symmetric PSK to codes on asymmetric PSK. They are also generalized to other types of noise than Gaussian, allowing more efficient block codes when noise is non-Gaussian. We provide examples of codes on asymmetric PSK that have higher minimum Euclidean distance than any comparable codes on symmetric PSK. Several classes of codes are shown to be optimal among codes on symmetric PSK since their Euclidean distance coincides with the bound.

The second paper considers a parallel computer system with \( m \) identical computers, where we study optimal performance precaution for one possible computer crash. We anticipate that some computer may crash, and restrict the cost in such a situation. How costly is such a precaution when no crash occurs? We set a restriction that the completion time of a parallel program after a crash is at most a factor \( r + 1 \) larger than if we use an optimal allocation with \( m - 1 \) computers. This is an \( r \)-dependent restriction of the set of allocations of a program. Then the worst-case ratio of the optimal \( r \)-dependent completion time in the case of no crash and the unrestricted optimal completion time defines a function \( f(r, m) \). In the paper we establish upper and lower bounds of the worst-case cost function \( f(r, m) \) and characterize worst-case programs.

The third paper considers the problem of Map Matching (MM), i.e. matching time and location measurements of a vehicle to a route in a road network. The paper presents a probabilistic algorithm for MM based on a second order hidden Markov model (HMM), as opposed to first order HMMs which are usually used. This allows a more detailed analysis of the data while preserving algorithmic complexity \( O(n) \). Both measurement densities and transition probabilities are determined with respect to Kolmogorov’s third axiom, which in this context implies that the probabilities are additive over a partition of a road segment.
Acknowledgments

First and foremost I would like to thank my supervisors Dr. Håkan Lennerstad and Dr. Magnus Nilsson for the help and encouragement they have given me during this work. I have learned a lot from discussions with both of them.

Furthermore, I wish to thank Prof. Lars Lundberg, Dr. Johan Holmgren and Dr. Mattias Dahl for fruitful cooperation with respect to parts of my research.

I would also like to thank Dr. Robert Nyqvist who have helped me greatly with \LaTeX.

My thanks also goes to all of my colleagues at BTH, in particular those at the department of mathematics and natural sciences, for being good friends and forming a fantastic research environment.

Last but not least I would like to thank my family for moral support.
Contents

Preface 1

1 Introduction 2
  1.1 Mathematical modeling 2
  1.2 Combinatorial optimization 2
  1.3 Complexity theory 3
  1.4 Some classical problems 5
  1.5 A couple of methods 7

2 Presentation of papers 11
  2.1 Paper I 11
  2.2 Paper II 12
  2.3 Paper III 13

List of Figures
  1 An illustration of matchings 9
Preface

This work consists of two parts. The first part introduces combinatorial optimization and presents the three papers which form the second part.

Work in the doctorate thesis:


Work in the licenciate thesis not in the doctorate thesis:


Other relevant work not in either thesis:


1 Introduction

1.1 Mathematical modeling

In this thesis combinatorial optimization problems are solved within three rather different application areas. The problems themselves are the result of mathematical modeling, which we understand as the process of finding precisely those mathematical problems that are relevant for the application area. If the model is too simple, then the solution to the resulting problem will be of limited value. But if it should be possible to make any progress towards solving the problem, then the model cannot be too complex. The task is thus not only to solve problems, it is to solve the right problems. From mathematical modeling point of view, problem formulation is as essential as problem solving. An example of the hazards of an inaccurate model is given in (Dantzig 1990).

The mathematical modeling can take many shapes, as it does in the three papers. It may start with a rather clear and constant application need, where the reformulations of the optimization problem are done entirely within the mathematics domain to provide yet stronger results for the application need. This is done in Paper I, where it results in sharper and more general bounds for a problem of combinatorial optimization. The basic formulation, laid down in (Nilsson and Lennerstad 2000), has allowed improvements and expansions, which can be seen as the source of both the licenciate thesis (Laksman 2010) and Paper I.

The modeling may also result in a distinct problem that from a mathematical point of view appears to be rather strange or artificial, but actually fits with the tools that are at hand for an engineer to fine-tune a system – as in Paper II. In such cases the problem formulation task may be as demanding as solving the problem itself.

Yet another form that the modeling may take is as an ongoing negotiation between theory and practice, which is characteristic of Paper III. Here the application demands that the mathematical model leads to a problem of low computational complexity, which must be negotiated with the demand for an accurate and theoretically sound model.

This introduction discusses ideas and tools in combinatorial optimization. Since the mathematical modeling is problem specific, it is again discussed in the presentations of the papers.

1.2 Combinatorial optimization

The topic of combinatorial optimization is a wide one, which has grown significantly since the advent of computers, and it can be described in different ways. We start by briefly describing two problems, which both have been and still are important for progress in combinatorial optimization. Both are described in more mathematical detail later.

Example 1 (Linear programming). Given is a set of $n$ different products and $m$ different raw materials. To produce one unit of product $i$ requires $a_{i,j}$ units of raw material $j$. We have only $b_j$ units of raw material $j$. The price of one unit of product $i$ is $c_i$. The income can be described as $c_1x_1 + \ldots + c_nx_n$, where $x_i$ is the number of produced units of product $i$. Which amounts of the different products are to be produced from the raw materials to maximize the income? The $x_i$:s are often integers, in which case it is an integer programming problem.

This linear programming problem also provides a mathematical formulation of an equally common and important cost-minimization problem. There are numerous applications of both problems in economics, telecommunications, scheduling and other engineering areas.

Example 2 (Traveling salesman problem (TSP)). A salesman is to visit $n$ cities. The distance between city $i$ and city $j$ is $c_{i,j}$. In which order should he visit them in order to minimize the total distance?
1.3 Complexity theory

The salesman may consider not only the distances between cities, but a more general cost function including other costs and issues that are relevant for the sales. There are numerous applications, where the “cities” may be replaced by costumers, DNA fragments, holes to be drilled in a metal plate, etc...

For some applications the descriptions above may fit very well. For other applications for these two examples, the mathematical problem can be developed and reformulated in different ways into more advanced combinatorial optimization problems, in order to produce results that are more exact and appropriate to those applications. In such cases, the formulations above are rather crude first approximations.

According to (Graham, Grötschel, and Lovász 1995), chapter 28.1, “There are basically two ways of presenting ‘combinatorial optimization’: by problems or by methods.” We attempt to go both ways. Section 1.4 contains an introduction to combinatorial optimization by a list of problems. The list is not comprehensive, but we have attempted to include problems of different types and different degrees of complexity. A brief introduction to complexity theory is given in Section 1.3. Section 1.5 describes a few fundamental methods for combinatorial optimization which are also of relevance to the papers in this thesis.

Before listing problems and discussing methods we turn to a more formal setting. Consider a function $f: S \mapsto \mathbb{R}$, usually called the objective function. Then an optimization problem is either a minimization problem

$$\min_{x \in S} f(x), \quad (1.1)$$

or a maximization problem

$$\max_{x \in S} f(x), \quad (1.2)$$

where $S$ is known as the feasible region. It is always possible to change a minimization problem into a maximization problem and vice versa by considering $-f$ instead of $f$. A particular optimization problem is specified by $f$ and $S$, and we have a combinatorial optimization problem when $S$ is discrete.

1.3 Complexity theory

How difficult is it to solve a certain problem with a computer, in terms of computer load? This is the central question under study in complexity theory. For a rigorous account of its concepts we refer to e.g. (Garey and Johnson 1979). We here only provide some basics that are needed for the theme of this thesis, and not always in a completely rigorous manner.

The complexity of a problem is quantified by comparing the minimum number of basic operations needed to solve it with the space needed to express it. We usually speak of “time” rather than “number of basic operations,” as the two are approximately proportional. To avoid handling all kinds of problems at once, complexity theory begins with decision problems. A decision problem is a problem with only two possible answers: “yes” and “no”. We are more interested in the complexity of optimization problems, so for the purpose of simplicity we restrict ourselves to decision problems of the form “Is $\{x \in S : f(x) \leq B\}$ nonempty?” or “Is $\{x \in S : f(x) \geq B\}$ nonempty?” for some specifications of $S, f$ and $B$. These have natural corresponding optimization problems: $\min_{x \in S} f(x)$ and $\max_{x \in S} f(x)$.

The space needed to express a problem of the form “Is $\{x \in S : f(x) \leq B\}$ nonempty?” is the space we need to specify $S, f$ and $B$ plus a constant. How much space is required to express a certain mathematical structure? We will get by with integers and graphs, and for “normal” languages, the space needed to express an integer $n$ is bounded by $p(\log(1 + |n|))$ for some polynomial $p$, while the space required to express a graph $G = (V, E)$ is bounded by $p(|V| + |E|)$ for some polynomial $p$.

That a problem $\Pi$ belongs to $P$ can briefly be explained as:
1. \( \Pi \) is a decision problem.

2. The answer can be determined in time which is bounded by some polynomial of the space needed to express \( \Pi \).

That a problem \( \Pi \) belongs to NP can briefly be explained as:

1. \( \Pi \) is a decision problem.

2. If the answer to \( \Pi \) is yes, then given a \( y \in \{ x \in S : f(x) \leq B \} \) it is possible to verify that \( y \in S \) and \( f(y) \leq B \) in time which is bounded by some polynomial of the space needed to express \( \Pi \).

Remark that the time needed to find \( y \) in point 2 above need not be bounded by any polynomial of the space needed to express the problem.

That a problem \( \Pi \) is NP-complete can briefly be explained as:

1. \( \Pi \) belongs to NP.

2. For every problem \( \Pi' \) in NP there is a transformation which can be performed in time which is bounded by a polynomial of the space needed to express \( \Pi' \) from \( \Pi' \) to \( \Pi \), such that \( \Pi' \) can be seen as a special case of \( \Pi \).

The immediate consequence of point 2 is that if any NP-complete problem belongs to \( P \), then all problems in NP must belong to \( P \). Whether or not such a problem exists is unknown. This is a famous unsolved problem: “Does \( P = NP \)?”

Consider a problem where the only mathematical structure that needs to be expressed is an integer. Such problems exist, for example “Is \( N \) nonprime?” requires only that a fixed length sentence and \( N \) are expressed. The space required to express such a problem is bounded by \( p(\log(1 + |N|)) \). An algorithm which requires \( N^\alpha \) for some \( \alpha > 0 \) time is thus not bounded by a polynomial of the space required to express the problem. An algorithm which is bounded by some polynomial of the maximum of the space needed to express the problem and the greatest integer needed to express the problem is said to be a pseudo-polynomial time algorithm. An NP-complete problem which can be solved by a pseudo-polynomial time algorithm is said to be NP-complete in the weak sense. Otherwise it is said to be NP-complete in the strong sense, unless \( P = NP \).

The concepts above concern only decision problems, but we are interested in optimization problems. Note that the solution to \( \min_{x \in S} f(x) \) also answers “Is \( \{ x \in S : f(x) \leq B \} \) nonempty?” so the optimization problem is at least as difficult as the corresponding decision problem. A problem is said to be NP-hard if it is at least as hard to solve as some NP-complete problem.

Consider an algorithm which for a given minimization problem and for some \( \rho > 0 \) outputs a solution \( y \) with

\[
    f(y) \in \left[ \min_{x \in S} f(x), (1 + \rho) \min_{x \in S} f(x) \right],
\]

or which for a given maximization problem and for some \( \rho \in (0, 1) \) outputs a solution \( y \) with

\[
    f(y) \in \left[ (1 - \rho) \min_{x \in S} f(x), \min_{x \in S} f(x) \right],
\]

and which does so in time bounded by a polynomial of the space required to express \( S \) and \( f \). Such an algorithm is a polynomial-time \( \rho \)-approximation algorithm. A sequence of such algorithms for all \( \rho > 0 \) is a polynomial-time approximation scheme (PTAS). If the algorithms in the scheme are also polynomial with respect to \( 1/\rho \), then it is known as a fully PTAS.
1.4 Some classical problems

In this subsection the descriptions of the problems focus on the mathematical nature of the problems, rather than the applications as was done in Example 1 and 2. The problems mentioned do not form an extensive list. The choices have been made to cover problems of different degrees of tractability as well as problems of distinct nature - covering problems on graphs, on sets and on integers.

For many combinatorial optimization problems, $S$ is finite. From a theoretical viewpoint such problems could be claimed to be trivial, as they can be solved by calculating the value of the objective function of each of the elements in the feasible region. From a practical viewpoint however, this approach is often intractable, as the feasible region can be too large for us to expect that a modern computer can manage such a task within reasonable time.

The problems below are described using integers, even in places where one may expect real numbers. As the complexity of a problem is considered to be a relation between time required to solve the problem and space required to express the problem, real numbers requiring infinite space to be expressed cannot be used. Instead of expressing a problem with real numbers, all numbers can be scaled so that very large integers correspond to smaller real numbers expressed with high resolution.

**Maximum cardinality matching** Consider a situation where a set of workers has to complete a certain number of tasks, where each task requires exactly one worker to be performed. Not all workers can manage any task, so a matching of tasks and workers is needed in order to get as many tasks as possible done. We may construct a graph where workers and tasks are vertices, and there is an edge between vertex $w$ and vertex $t$ if $w$ represents a worker capable of performing task $t$. This gives the following optimization problem:

**Problem 3.** Consider a graph $G = (V, E)$. The problem is to find $\max_{X \in S} f(X)$ where $f(X)$ is the number of elements in $X$ and

$$S = \{X \subseteq E : (u, v \in X, u \neq v) \Rightarrow (u_i \neq v_j, \forall i, j \in \{1, 2\})\}.$$

In the application mentioned, $X$ is a subset of connections between workers and tasks such that no task has two workers and no worker two tasks.

The first polynomial time algorithm for solving the problem on a non-bipartite graph was given in (Edmonds 1965), and requires $O(|V|^4)$ operations. Subsequent improvements of Edmonds’ algorithm have been made, for example the algorithm of (Micali and Vazirani 1980) requires only $O(\sqrt{|V||E|})$ operations.

A matching on a graph is a subset of the edges such that no two edges in the subset are incident to the same node. The set $S$ is the set of all matchings. There are other matching problems using subsets of $S$ as feasible region. For further reading on matching problems, see (Lawler 1976; Lovász and Plummer 2009).

**Traveling salesman problem** We have already seen a natural application of TSP in Example 2.

**Problem 4.** Consider a set $V = \{v_1, \ldots, v_n\}$ and a symmetric function $c : V \times V \rightarrow \mathbb{Z}_+$. The problem is to find $\min_{x \in S} f(x)$ where $S$ is the set of permutations $(x_1, \ldots, x_n)$ of $V$ and $f(x) = c(x_n, x_1) + \sum_{i=1}^{n-1} c(x_i, x_{i+1})$.

TSP is known to be NP-hard, where the corresponding decision problem is known to be NP-complete in the strong sense. The asymptotically fastest exact algorithm known was presented in (Held and Karp 1962) and requires $O(n^2 2^n)$ operations. Furthermore, for general TSP, there is no $\rho > 0$ for which a polynomial-time $\rho$-approximation algorithm exist, unless $P = NP$ (Sahni
and Gonzales 1976). However, if the cost function $c$ is restricted to satisfy the triangle inequality, then polynomial-time approximation algorithms exist, e.g., (Christofides 1976). If $V$ is a set of points in $\mathbb{R}^d$ and the cost-function is Euclidean distance, then even PTAS exist (Arora 1998). For further reading on TSP, see (Lawler et al. 1985).

**Problems with linear conditions and objective functions** Here we formulate a mathematical description of the problem encountered in Example 1. The feasible region determines whether it is a linear programming problem (LP), an integer programming problem (IP) or a binary integer programming problem (BIP). Note that LP and IP are the only problems discussed in this thesis where the feasible region may be infinite.

**Problem 5.** Let $f(x) = \sum_{i=1}^{n} c_i x_i$ and

- $S = \{x \in \mathbb{R}^n : \sum_{i=1}^{n} a_{i,j} x_i \leq b_j, \forall j \in \{1, \ldots, m\}\}$, \hspace{1cm} \text{(LP)}
- $S = \{x \in \mathbb{Z}^n : \sum_{i=1}^{n} a_{i,j} x_i \leq b_j, \forall j \in \{1, \ldots, m\}\}$, \hspace{1cm} \text{(IP)}
- $S = \{x \in \{0, 1\}^n : \sum_{i=1}^{n} a_{i,j} x_i \leq b_j, \forall j \in \{1, \ldots, m\}\}$, \hspace{1cm} \text{(BIP)}

for some integers $a_{i,j}, b_j, c_i, i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\}$.

The problem is either to find $\min_{x \in S} f(x)$ or to find $\max_{x \in S} f(x)$.

Linear programming is a continuous optimization problem. The feasible region forms a polyhedron, and when the objective function is linear an optimal point can be found in a corner – provided that a feasible point exists and the optimum is finite. One can see it as a problem of determining which of the $m$ conditions that should be satisfied with equality, and in this respect it is a combinatorial problem. LP is often solved by the simplex method (Dantzig 1963), starting in some corner of the polyhedron, trying to reach the optimal corner by moving along the edges. Simplex is very swift in practice and usually needs to examine only between $2m$ and $3m$ corners in the polyhedron (Dantzig 1963; Hoffman et al. 1953). The worst-case performance of the simplex algorithm is however not polynomial (Klee and Minty 1972). Polynomial time algorithms exist for LP (Khachiyan 1979; Karmarkar 1984).

Integer and binary programming are NP-hard. Some of the methods for solving these problems include solving linear programming problems obtained by relaxing the restriction $x \in \mathbb{Z}^n$, or $x \in \{0, 1\}^n$, to $x \in \mathbb{R}^n$.

Further reading on linear and integer programming can be found in e.g., (Schrijver 1998).

**Multiprocessor scheduling** Consider a computer with several identical processors. The computer is supposed to complete a program consisting of a set of processes. Each process requires a certain amount of time to be completed on any of the processors. We also assume that a process may never be transferred from the processor where it is started, and that the processes are independent. How should the processes be allocated on the processors to minimize the time until all processes are done? Let $m$ be the number of processors and $n$ be the number of processes. Let $p_i$ be the time needed for one processor to complete process $i$, and let $a_i$ be the processor to which process $i$ is allocated, and let $A = (a_1, \ldots, a_n)$ be an allocation. The problem can be formulated mathematically as:
1.5 A couple of methods

**Problem 6.** Let \( m, n \in \mathbb{Z}_+ \) and consider a vector \( P = (p_1, \ldots, p_n) \in \mathbb{Z}_+^n \). Let \( A = \{1, \ldots, m\}^n \), let \( S = A \) be the feasible region and let

\[
f(A) = \max_{1 \leq i \leq m} \sum_{k: a_k = i} p_k
\]

be the objective function. The multiprocessor scheduling problem is to find \( \min_{A \in S} f(A) \).

The corresponding decision problem is mentioned in (Garey and Johnson 1979), where it is shown to be NP-complete in the strong sense. From the NP-completeness of the decision problem it follows that the optimization problem is NP-hard.

**Knapsack problem** We have a set of objects, each of which has a value and a weight. We wish to pack as much value as possible in a knapsack, but there is a maximum weight that we can carry. Which objects should we pack? This problem may be formulated as:

**Problem 7.** Consider a set \( A = \{a_1, \ldots, a_n\} \), two functions \( v, w : A \to \mathbb{Z}_+ \) and a number \( B \in \mathbb{Z}_+ \). The problem is to find \( \max_{X \in S} f(X) \) where \( f(X) = \sum_{a \in X} v(a) \) and

\[
S = \left\{ X \subseteq A : \sum_{a \in X} w(a) \leq B \right\}.
\]

The knapsack problem is known to be NP-hard, but the corresponding decision problem is only NP-complete in the weak sense. In particular, an algorithm solving the knapsack problem in \( O(nB) \) basic operations is known. Also fully PTAS exist (Ibarra and Kim 1975).

For further reading on the knapsack problem, see (Kellerer, Pferschy, and Pisinger 2004).

1.5 A couple of methods

We here give brief descriptions of two methods for combinatorial optimization; local improvement, which plays a part in Paper II, and of dynamic programming, which is implicitly used through the entire Paper III. A cover of several methods can be found in *e.g.* in chapter 28 of (Graham, Grötschel, and Lovász 1995).

**Local Improvement:** When faced with a minimization problem, we wish to find the global minimum, *i.e.* \( x \in S \) such that \( f(x) < f(y) \) for all \( y \in S \). A neighborhood function is a function \( N : S \to \mathcal{P}(S) \), where \( \mathcal{P}(S) \) denote the power set of \( S \), *i.e.* the set of all subsets of \( S \). A local minimum is a point \( x \in S \) such that \( f(x) < f(y) \) for all \( y \) in some neighborhood \( N(x) \) of \( x \). If no local optima exist, except the global optima, then finding a local optimum is sufficient for solving the optimization problem.

We can find a local optimum by repeatedly improving the currently best known solution by some small perturbation of it. Note that a feasible point is required to start local improvement. Some problems have obvious feasible points which can be used, such as the empty set in Problem 3 and 7. For other problems, such as Problem 5, finding a feasible point may be nontrivial.

Say that we wish to solve a minimization problem, expressed as (1.1). The method of local improvement can be described algorithmically as:

1. Input: a neighborhood function \( N \) and a feasible point \( x \in S \).
2. Try to find a \( y \in N(x) \) such that \( f(y) < f(x) \).
3. If a $y$ was found in step 2, let $x := y$, and go to step 2. If no $y$ was found in step 2, report $x$ as a local optimum with respect to $N$ and terminate.

When there are several $y \in N(x)$ such that $f(y) < f(x)$, how to choose which such $y$ that should replace $x$ in the next iteration may be important and is problem specific. The choice can affect both which local optimum is found and how fast a local optimum is found.

There may be $x \in S$ such that $\{y \in N(x) : f(y) < f(x)\}$ is empty but $\{y \in N(x) : f(y) = f(x)\}$ contains more than one point. In such a case $\{y \in N(x) : f(y) = f(x)\}$ forms a plateau. In some cases one will not “break” the algorithm as soon as a plateau is reached, but rather move around on the plateau in search of a good continuation. This doesn’t concern us much, for the way in which we use local improvement in Paper II is by noting that a point cannot be a global optimum if it can be locally improved.

A lack of non-global optima does not guarantee that local improvement is efficient. Linear programming, see Problem 5, is often dealt with by the simplex algorithm. The optimum is known to lie in a corner in a polyhedron. In the simplex algorithm, the neighborhood of a corner $x$ is formed by all corners which can be reached from $x$ by moving along one edge. In LP all local optima are global optima, so simplex finds a global optimum if one exists, but this may require exponential time (Klee and Minty 1972). If there are many optimal solutions, which then form a simplex of dimension at least 1, the simplex algorithm only finds one or several corners in this solution simplex. Furthermore, if the next corner choice in the simplex algorithm is chosen using maximal improvement of the goal function, this may lead to a larger number of steps of the algorithm than if a different neighbor is chosen.

In the simplex algorithm we saw that the neighborhood was taken to be corners which are close with respect to which constraints are fulfilled with equality. Closeness, with respect to some distance, is often used to define neighborhoods also for other problems. Let $d$ be some distance measure on $S$, i.e. a function $d : S \times S \rightarrow \mathbb{R}_+$ with the properties

- $d(s_1, s_2) = 0$ if and only if $s_1 = s_2$ and
- $d(s_1, s_2) = d(s_2, s_1)$.

A natural way to define a neighborhood $N$ is as all feasible points which are sufficiently close with respect to $d$:

$$N(x) = \{s \in S : d(x, s) \leq c\}$$

(1.3)

for some $c > 0$.

How a neighborhood is defined is of essence when it comes to local improvement. We exemplify by maximum cardinality matching, Problem 3. The feasible region $S$ consists of sets of pairwise non-adjacent edges. Let $d$ be the size of the symmetric difference,

$$d(s_1, s_2) = |s_1 \cup s_2| - |s_1 \cap s_2|,$$

and define neighborhoods by (1.3) with $c = 1$. Since it is a maximization problem the objective function has to increase in every iteration of local improvement. Thus edges will be added to the solution one by one until no more edges can be added. A local optimum is illustrated in Figure 1(a). Matchings which are locally optimal with respect to this particular neighborhood are said to be maximal matchings. Note that had a neighborhood been defined with $c = 3$, then the matching in Figure 1(a) would not be a local optimum, and local improvements would be performed until the global optimum – illustrated in Figure 1(b) – would be found. The downside of using $c = 3$ is that running step 2 of the algorithm would have higher complexity than with $c = 1$.

To conclude we can say that if neighborhoods are small it is easy to get stuck in a local minimum. Larger neighborhoods reduces this risk, but usually leads to a more complex algorithm. When the neighborhoods used are very large or have a complex structure the method is usually not termed local optimization.
1.5 A couple of methods

**Dynamic Programming:** Dynamic programming is applicable when a problem can easily be solved from subproblems and when the total number of subproblems which need to be solved is small.

We illustrate by two examples where dynamic programming can be applied with different degrees of success.

Our first example is the knapsack problem, Problem 7, where dynamic programming results in an algorithm with pseudo-polynomial complexity. We are to fill a knapsack – which can hold weight \( B \) – with objects from the set \( A = \{a_1, \ldots, a_n\} \), where \( w(a_i) \) is the weight of \( a_i \). We wish to pack the knapsack with as much value as possible, where \( v(a_i) \) is the value of \( a_i \). A knapsack problem is fully defined by \( A, B, v \) and \( w \). Denote the optimal value of such a knapsack problem \( K(B, A) \).

We denote \( \{a_1, \ldots, a_k\} \) by \( A_k \). Observe that

\[
K(C, A_k) = \max \left( K(C, A_{k-1}), v(a_k) + K(C - w(a_k), A_{k-1}) \right).
\]

If \( K(C, A_{k-1}) \) is known for all \( C \), then \( K(C', A_k) \) can for any \( C' \) be calculated by only one addition and one comparison. In total we need to solve at most \( O(nB) \) knapsack problems, each of which are solved by \( O(1) \) basic operations.

Our second example is TSP, Problem 4, to find a route visiting each element in \( V = \{v_1, \ldots, v_n\} \) of lowest cost, where the cost of traveling between \( v_i \) and \( v_j \) is \( c(v_i, v_j) \).

We view TSP as the problem of finding the route of lowest cost from \( v_1 \) to \( v_1 \) visiting each element in \( V \setminus v_1 \) exactly once. A subproblem is to find the route of lowest cost from \( v_1 \) to \( v_i \), for \( i \neq 1 \), visiting each of \( k \) of the elements in \( V \setminus \{v_1, v_i\} \) exactly once. The set of elements to be visited on such a route is an element from

\[
\mathcal{V}(k, i) = \{X \subseteq V \setminus \{v_1, v_i\} : |X| = k\}.
\]

A subproblem is completely determined by \( c, v_1, W \in \mathcal{V}(k, i) \) and \( v_i \). We denote the optimal value of such a subproblem \( TSP(W, v_i) \).

Observe that if \( W \in \mathcal{V}(k, i) \), then

\[
TSP(W, v_i) = \min_{w \in W} TSP(W \setminus w, w) + c(w, v_i)
\]

The TSP we wish to solve is \( TSP(V \setminus v_1, v_1) \). We note that

\[
\sum_{v_i \in V} |\mathcal{V}(k, v_i)| = (k + 1) \binom{n - 1}{k + 1},
\]
In the total number of subproblems we need to solve is

$$\sum_{k=0}^{n-2} \sum_{v_i \in V} |V(k, v_i)| = \sum_{k=0}^{n-2} (k + 1) \binom{n-1}{k+1} = (n-1)2^{n-2}.$$  

To perform the necessary additions and comparisons, the number of operations needed to solve TSP by dynamic programming as described above are $O(n^2 2^n)$. Dynamic programming leads to the best algorithm known to solve TSP exactly, but the complexity is still exponential in $n$. This use of dynamic programming for TSP was first described in (Held and Karp 1962).

Our description of dynamic programming, solving a problem by solving subproblems, doesn’t differ from the method known as divide and conquer, which is covered in any basic literature on algorithm design such as (Alsuwaiyel 1999). The difference lies in the size of the subproblems. Divide the subproblems into “levels,” so that a subproblem on level $k$ is solved by subproblems on level $k-1$. In divide and conquer the size of the subproblems grow exponentially with the level. In dynamic programming the size of the subproblems grow linearly with respect to the level.

The linear growth of the size of subproblems lends itself well to an optional view on dynamic programming. It can be viewed as a decision process, where we examine possible decisions breadth first. In the knapsack problem we make the $n$ decisions: “Should $a_k$ be packed into the knapsack?” for all $k \in \{1, \ldots, n\}$. In TSP we make the $n$ decisions: “Which vertex should be the $k$:th vertex in the cycle?” for all $k \in \{2, \ldots, n\}$. The number of decisions are kept down by disregarding decisions which cannot lead to feasible solutions or which are clearly suboptimal.

For a thorough cover of dynamic programming, see (Bellman 2003).
2 Presentation of papers

2.1 Paper I

Paper I is antedated by a series of papers, (Nilsson and Lennerstad 2000; Nilsson and Lennerstad 2005; Nilsson, Lennerstad, and Laksman 2008; Laksman, Lennerstad, and Nilsson 2010; Laksman, Lennerstad, and Nilsson 2009). The previous results are generalized in two directions; Paper I considers asymmetric phase shift keying (APSK), where the earlier papers considered symmetric phase shift keying (SPSK) – which may be viewed a special case of APSK – and it considers additive white noise where earlier papers considered additive white Gaussian noise.

One way of measuring the performance of a code is by the minimum distance between its codewords. As a received signal is corrected to the closest codeword, it is important that the distance between two signals must be monotonically decreasing with the probability that the communication channel transforms one of the signals to the other. Correcting signals to the nearest codeword will then correspond to maximum-likelihood correction, and any error smaller than half the minimum distance will be successfully corrected.

Consider an abelian group \( G \) and a finite set \( \Sigma \subseteq G \) with \( q = |\Sigma| \), where \( \Sigma \) is the set of symbols used for communication. Let \( n \) be the number of symbols in a word, and let \( c \) be the number of codewords. We have \( q, n, c \in \mathbb{Z}_+ \) with \( c \leq q^n \). Consider an additive and translation-invariant distance measure \( d \) on \( G \), i.e., a function \( d : G \times G \rightarrow \mathbb{R} \) which satisfies

\[
d(x, y) = d(x + z, y + z),
\]

(translation invariance)

for any \( x, y, z \in G \), and which extends to

\[
d(x, y) = \sum_{i=1}^{k} d(x_i, y_i),
\]

(additivity)

for any positive integer \( k \). Let the feasible region \( S \) be defined as

\[
S = \{ C \subseteq \Sigma^n : |C| = c \},
\]

i.e. the set of all sufficiently large codes. Define the objective function \( f \) as

\[
f(C) = \min_{x, y \in C, x \neq y} d(x, y),
\]

i.e. the minimum distance between codewords. The problem we would like to solve is \( \max_{C \in S} f(C) \).

The necessary technical background can be found in Paper I. A somewhat extended introduction can be found in (Laksman 2010). A more thorough technical background can be found in most literature on coding theory, such as (Berlekamp 1968) or (Roth 2006).

So far the model is straightforward, direct from the application. In order to form an upper bound on \( \max_{C \in S} f(C) \), we need to remodel the problem. The model used was introduced by Elias, and is described in (Berlekamp 1968). It consists of forming \( q^n \) spheres, centered at each of the elements in \( \Sigma^n \). By use of the pigeon hole principle, at least one sphere must contain at least \( K \) codewords, for some \( K \) determined by \( q, n, c \) and the size of the spheres. Pick such a sphere with at least \( K \) codewords and call the set of codewords in the sphere \( W \). If the spheres are sufficiently large, we get \( K \geq 2 \). Then the following is used

\[
f(C) = \min_{x, y \in C, x \neq y} d(x, y) \leq \min_{x, y \in W, x \neq y} d(x, y) \leq \frac{\sum_{x, y \in W, x \neq y} d(x, y)}{K(K-1)}.
\]

Originally the spheres were defined by the distance measure used to measure performance of the codes. This was generalized in (Nilsson and Lennerstad 2000) where the spheres where defined...
according to another distance measure. The distance measure used was shown to give better bounds when $c/q^n$ is large.

All of (Nilsson and Lennerstad 2005; Nilsson, Lennerstad, and Laksman 2008; Laksman, Lennerstad, and Nilsson 2010; Laksman, Lennerstad, and Nilsson 2009) deal with the problem of which distance measure to use to define the spheres so that the bound,

$$\frac{\sum_{x,y\in W, x\neq y} d(x,y)}{K(K-1)},$$

is minimized – for a specific distance measure $d$ and some restrictions on the symbol sets. For this model, a sphere can be defined as a discrete set of points, but it can also be defined via the set of additive and translation-invariant distance measures, which leads to a continuous optimization problem. By modeling the problem as continuous, we rewrite it as a set of linear programming problems in Paper I. One result is that if the distance measure used to measure the performance of the codes, $d$, is a metric, then the best bound is obtained by defining the spheres by the same distance measure.

### 2.2 Paper II

The second paper concerns the scheduling of independent processes on a cluster of computers, and the possible need for rescheduling in case one of the computers breaks down. We consider a situation where there is a deadline for when all processes must be finished, but where we would still like to finish all processes as soon as possible. To get a manageable model, we assume that

- at most one computer crash will take place,
- processes are not reallocated unless the computer they are allocated to crashes,
- a computer which receives additional processes after a reallocation has not lost any time being idle.

The assumptions are strong, but not without reason. For example, if computer crashes are very rare, then the probability of more than one crash will be much lower than the probability of one crash. If the progress made so far can be saved somewhere, then the worst case is a crash immediately after start, in which case no computer is idle before receiving new processes. Other models are certainly conceivable, but we have chosen a model which is manageable and which we believe is somewhat relevant.

The problem of Paper II is similar to Problem 6, but an additional constraint is introduced in the feasible region. We choose the terminology “a cluster of computers” rather than “a computer with multiprocessors,” as is the terminology in Problem 6. The reason is that the crash of a processor in computer with multiple processors cause the entire computer to go down, which is not the case for independent computers.

Let $m$ be the number of computers and $n$ be the number of processes, so that $m, n \in \mathbb{Z}_+$. A program $P$ is defined by a vector $P = (p_1, \ldots, p_n) \in \mathbb{R}_+^n$, where $p_i$ is the time required to complete process $i$. Note that in Problem 6 we have integer values, which is necessary only if we intend to have precise discussions regarding complexity. Let $A = \{1, \ldots, m\}$ be the set of all allocations, where $A = (a_1, \ldots, a_n) \in A$ says that process $i$ is to be run on computer $a_i$. Let $A_P = \{A_1, \ldots, A_m\}$, where $A_i = \{p_j : a_j = i\}$ is the set of running-times for the processes to be run on computer $i$. Let $A'(A, j)$ be the set of allocations that we can attain from allocation $A$ when the processes allocated to computer $j$ need to be reallocated to other computers. Formally, $A'(A, j)$ is the subset of $A$ such that for any element $B = (b_1, \ldots, b_n)$ in $A'(A, j)$, exactly one of the statements $a_i = b_i$ and $a_i = j$ is true. Let $T(A_P)$ denote the time required to run program
Next, let $\iota$ be the “worst” computer which can crash,

$$\iota = \arg\max_j \min_{B \in A'(A,j)} T(B_P),$$

and let $A'$ be the best possible reallocation of $A$ when computer $\iota$ crashes,

$$A' = \arg\min_{B \in A'(A,\iota)} T(B_P).$$

Further, let $U$ be the best allocations in the case of no crash,

$$U = \{A \in A : T(A_P) = \min_{B \in A} T(B_P)\},$$

and let $\Phi$ be the allocation in $U$ which is best in case of a crash,

$$\Phi = \arg\min_U T(A'_P).$$

Let $V$ be the best allocations in the case the worst possible computer crashes,

$$V = \{A \in A : T(A'_P) = \min_{B \in A} T(B'_P)\},$$

and let $\Psi$ be the allocation in $V$ which is best in case of no crash,

$$\Psi = \arg\min_V T(A_P).$$

Note that $A'$, $\Phi$ or $\Psi$ may not be unique.

Let the feasible region be

$$S = \{A \in A : T(A'_P) \leq (1 + r)T(\Psi'_P)\}$$

for some specified real number $r \in [0, 1]$. Let $f(A) = T(A_P)/T(\Phi_P)$ be the objective function. What is asked for is then $\min_{A \in S} f(A)$. Compared to Problem 6 the changes are a more restricted feasible region and a scaling factor $1/T(\Phi_P)$ in the objective function.

It is fairly straightforward to show that this optimization problem is NP-hard, see the concluding section of Paper II. Rather than attempting to find an explicit solution to the problem, Paper II considers $P$ such that the optimal value of the minimization problem is maximized, and hence determines an upper bound on the minimization problem. We see that a continuous maximization problem with respect to $P$ is involved. This maximization problem is not solved, but a few properties which $P$ must have in order to be optimal are determined. This is done by local optimization, i.e. by noting that if $P$ lack certain properties it can be locally improved. The properties of such a program $P$ allow the formulation of certain inequalities. From these inequalities we calculate an upper bound on the maximization problem (with respect to $P$) which is also an upper bound on the minimization problem (with respect to $A$). By considering some explicit $P$, the bound is shown to be tight for many $(P, m, r)$, and for many other $(P, m, r)$ the bound is shown to be near the optimal such bound.

2.3 Paper III

The third paper concerns the problem of map matching (MM). A very concrete recent application for MM is a vehicle which travels along roads in a network. The position of the vehicle is measured at some discrete set of points in time (e.g. by GPS) and the measurements are disturbed
by noise. For the purpose of real-time navigation assistance, MM concerns determining the current position of the vehicle, which must be updated every time a new measurement is made. For other applications, such as electronic toll collection, MM concerns determining the route travelled by the vehicle when a series of measurements have been sampled.

Due to unpredictable noise, absolute certainty about the route travelled cannot be achieved. One has to seek the most likely route, leading to a specific optimization problem, to be described. We divide the problem into two steps:

1. One road is assigned to each measurement. Note that one road may be assigned to more than one measurement, so MM is not a matching problem in the same sense as Problem 3.

2. Where the roads found in the previous step do not form a complete route, additional roads must be included until a complete route is formed.

The second step is of course superfluous for the purpose of real-time navigation. Remark that while common, such a two-step approach is not necessary; another way of modeling the problem is to construct a curve from the measurements and attempt to recognize the curve as a set of roads in the road network, which becomes a form of pattern recognition. Such a model leads to a one step approach, and may be found in e.g. (Brakatsoulas et al. 2005).

Paper III focuses solely on the first step. For comparability of results, the second step is handled by filling in gaps from the first step by the shortest route, which is a natural and simple approach, but need not be the most adequate one.

Let \( \mathcal{R} = \{r_1, \ldots, r_m\} \) be the space of roads, each road leading from intersection to intersection. Let \( \mathcal{Z} \) be all positions on earth and let \( \mathbf{z} \in (\mathbb{R}_+ \times \mathcal{Z})^n \) be the \( n \) measurements, where the first coordinate represents time. Define the feasible region as \( S = \mathcal{R}^n \). We consider the “best” route to be the most likely route, i.e. \( \arg \max_{\mathbf{r} \in S} P(\mathbf{r}|\mathbf{z}) \), so that \( P(\mathbf{r}|\mathbf{z}) \) plays the role of the objective function. However, we have yet to define \( P(\mathbf{r}|\mathbf{z}) \) mathematically. We settle for some model of the real world which gives us a way to calculate \( P(\mathbf{r}|\mathbf{z}) \). A model close to reality ensures that few erroneous matchings are made, but high complexity may make the solution obsolete when calculated.

For the purpose of real-time navigation assistance, each time a new measurement is made the current position must be updated in constant time. The restriction on complexity is not as clear cut for applications where we must determine a complete route from a series of measurements. However, complexity worse than \( O(n) \) would force us to treat long routes as a sequence of short routes. Viterbi’s algorithm (Viterbi 1967), which is an instance of dynamic programming, manages both to update the most likely position in constant time and to give the most likely route using \( O(n) \) time.

Viterbi’s algorithm has been used for MM on several occasions, where \( P(\mathbf{r}|\mathbf{z}) \) has been determined by modeling driving behaviour by a first order hidden Markov model (HMM), and by modeling noise as bivariate additive white gaussian noise.

One contribution of Paper III is that \( P(\mathbf{r}|\mathbf{z}) \) has been determined by modeling driving behaviour by a second order HMM and by modeling noise as a bivariate AR(1)-process. This permits the model to be closer to reality, at the cost of computational complexity. Note that higher order of the HMM does not increase the complexity in terms of \( n \), but slows down the algorithm none the less.

A second contribution of Paper III is the observation that measurement densities and transition probabilities – the building blocks of Viterbi’s algorithm – should satisfy the basic rules of probability theory. Special attention is paid to Kolmogorov’s third axiom, which is shown to have been violated by numerous models which have been used. Naturally, a probabilistic model which satisfies the rules of probability theory can be expected to be closer to reality than one that doesn’t.
To validate the suggested approach, Paper III also presents the results of an implementation of an algorithm based on a second order HMM where respect has been paid to Kolmogorov’s third axiom when determining measurement densities and transition probabilities. The implementation is very basic, and no optimization of the parameters in the model has been made. The algorithm is shown to perform approximately as well as leading MM algorithms, but is not shown to perform better.
References


— (2005). “Improved Upper Bound on the Minimum Euclidean Distance for Block Coded Phase Shift Keying”. In: *RVK05, Linköping, Sweden*. 


Generalized Upper Bounds on the Minimum Distance of PSK Block Codes

Efraim Laksman, Håkan Lennerstad, Magnus Nilsson
Generalized Upper Bounds on the Minimum Distance of PSK Block Codes

EFRAIM LAKSMAN†, HÅKAN LENNERSTAD† AND MAGNUS NILSSON‡
†School of Engineering, Blekinge Institute of Technology, Campus Gräsvik, Valhallavägen 1, Karlskrona, Sweden.
‡Linnaeus University, Kalmar Maritime Academy, Nygatan 18 B, Kalmar, Sweden.
[Received on 2 October 2012]

This paper generalizes previous optimal upper bounds on the minimum Euclidean distance for PSK block codes, that are explicit in three parameters: alphabet size, block length and code size. The bounds are generalized from codes over symmetric PSK to codes over asymmetric PSK and also to general alphabet size. Furthermore, block codes are optimized in presence of other types of noise than Gaussian, which induces also non-Euclidean distance measures. Examples of codes over asymmetric PSK are provided that have higher minimum Euclidean distance than any codes over symmetric PSK, and several classes of codes are shown to be optimal among codes over symmetric PSK. The underlying geometric approach, based on Elias’ sphere argument, is also systemized in several ways.

Keywords: asymmetric PSK, symmetric PSK, Elias bound.

1. Introduction

Phase Shift Keying (PSK) is a commonly used signal constellation in data transmission and communication systems, where often Euclidean distance is the preferred distance measure. The present paper improves in several ways previously published bounds on the minimum Euclidean distance valid for any symmetric PSK with alphabet size \( q \), word size \( n \) and code size \( |C| \).

The PSK signal constellation corresponds to a rather fundamental mathematical-geometrical optimization problem, in finding the minimum distance of a set of points in a universe equipped with a specific multidimensional metric. This paper belongs to a line of research that develops the underlying geometrical problem by extending Elias’ sphere argument in different ways. We claim that the approach used allows a larger flexibility in how performance questions may be stated, investigated and improved, as for example being useful for any distance measure – in this report we go beyond Euclidean distance measure. A different distance measure simply gives different coefficients in the optimization. In this paper we present some improvements of this extension of Elias’ sphere argument alongside with the main results, which are generalizations and improvements of upper bounds for the minimum Euclidean distance.

Strictly, PSK means only that the signals used differ in phase, but neither in amplitude nor frequency. If the phases used are evenly spread on the interval \((-\pi, \pi]\), i.e. the symbols are evenly spread on the unit circle, we will refer to this constellation as symmetric PSK; as is common in the literature. Some research has been done on signal constellations where the symbols are unevenly spread on the unit circle. In this paper we consider codes which are geometrically uniform (GU), meaning that the set of distances from one symbol to the others is the same for all symbols (see Definition 2.1); which we refer to as asymmetric PSK (sometimes the term “unbalanced” is used). Note that the set of codes over
symmetric PSK is a subset of the set of codes over asymmetric PSK. The two are illustrated in Figure 1.

Elias has formed a method to find upper bounds on the minimum distance of block codes with GU signal constellations, where minimum distance is measured by an additive, translation-invariant distance measure. This method has been used for several signal constellations, in particular for symmetric PSK, see Piret (1986). Note that for PSK, the distance of interest is often squared Euclidean distance, since when assuming additive white Gaussian noise (AWGN), finding the codeword closest to a signal will correspond to maximum likelihood (ML) decoding.

The method has been extended by the present authors in a series of reports (see Section 3). In the extended results the bounds are explicit in the parameters block length, alphabet size and code size, and are tight in many cases, i.e., show several code classes to be optimal with respect to minimum squared Euclidean distance.

The extension of Elias’ sphere argument for forming upper bounds on codes works when applied to block codes over asymmetric PSK with additive and white noise; note that this noise does not need to be Gaussian. This establishes previous results for symmetric PSK with AWGN in a far more general setting (Theorem 4.3, Theorem 4.4, Theorem 4.5), as well as yields some qualitative results which are more than just generalizations of previous results (Theorem 4.1, Corollary 4.1, Theorem 4.2, Corollary 4.2). Secondly, a theorem which cannot be generalized with respect to noise is generalized to the case of asymmetric PSK, while maintaining the restriction of AWGN (Theorem 4.7).

For the relevance of the bound, some examples (Examples 5.1, 5.2) are given of optimal block codes over asymmetric PSK. These codes have higher minimum Euclidean distance than optimal block codes over symmetric PSK, where both codes have the same block length, code size and alphabet size. Results of higher minimum Euclidean distance by use of asymmetric PSK have earlier been presented for PSK trellis codes Divsalar et al. (1987); Stüber & Katz (1995), and for block codes restricted to a special class of multi-level codes Bali & Rajan (1998). As far as the present authors know this is the first time such results are presented for general PSK block codes, i.e., without restriction to some particular code construction. The codes in Examples 5.1, 5.2 are rather short, so for purposes of communication, the codes presented in Bali & Rajan (1998) may be more relevant.
Asymmetric modulation has also been discussed for other purposes than achieving high minimum Euclidean distance. In Isaka et al. (2000) asymmetric modulation is used for unequal error protection and in Zhang et al. (2009) asymmetric modulation is used for minimizing the bit error rate.

Bounds for minimum Euclidean distance of codes over asymmetric PSK, which are asymptotic in $n$ are described in Viswanath (2004); Rajan & Viswanath (2003), while the present bound is explicit in $n$. The assumption of Euclidean distance is motivated by the AWGN-channel assumption.

An asymptotic bound on the minimum distance of codes over asymmetric PSK (without the restriction that the signal constellation should be GU) was presented in Wyner (1987).

The following improvements are presented for the bound on the minimum Euclidean distance for PSK coding, compared with previous work (Nilsson & Lennerstad (2000, 2005); Nilsson et al. (2008); Laksmman et al. (2009a,b, 2010)):

1. Bounds are generalized to codes over more general modulation: from symmetric PSK to also include asymmetric PSK (Theorem 4.4).
2. The bound is tightened in Theorem 4.5 by using the bound of Theorem 4.4 as an upper limit and exploiting symmetry properties.
3. For all types of PSK considered, the bounds are generalized from squared Euclidean distance to additive translation-invariant distance measures (Theorems 4.4 and 4.5). This provides an upper bound on the minimum distance for codes that more efficiently exploit the statistical properties of the noise.
4. Specific codes over asymmetric PSK are provided that have higher minimum Euclidean distance than any comparables code over symmetric PSK, although they are codes of very short block lengths (Example 5.1 and Example 5.2).
5. In the set of codes over symmetric PSK, we demonstrate for certain code classes that they are optimal in the sense that they have highest possible minimum Euclidean distance with given parameter values $q$, $n$ and $|C|$, since their minimum Euclidean distance in these cases coincides with the bound (see Examples 5.3 – 5.7).

This means that results 1 and 3 are generalizations of the bound in previous papers, while 2 is an improvement of the bound. The results 4 and 5 demonstrate the relation between concrete codes and the bound.

Elias’ sphere argument has been modified by allowing a different distance measure in the critical sphere – a so-called inner distance measure. It has turned out that a tighter bound on the Euclidean distance measure – the outer distance measure – can be derived by optimizing the inner distance measure. As Elias did, we regard the set of codewords in the critical sphere as rows in a matrix, and then switch to column view. Finding extremal codeword sets then translates into finding extremal columns.

This paper contributes to the extension of Elias’ sphere argument with the following results:

1. If the outer distance measure is a metric, then the inner distance measure which yields the best bound is the outer distance measure (Corollary 4.1).
2. We present simple conditions to check whether a generalized $p$-Gaussian induced distance measure (see Definition 2.3 and the comments following it) is a metric on a PSK signal constellation. We get one condition for asymmetric PSK (Theorem 4.2) and an easier one for symmetric PSK (Corollary 4.2).
3. For an extremal column $x$, we always have $d_E^2(x) \leq 4$ (Theorem 4.7). This simple and general condition reduces dramatically the set of candidate columns.

In Section 2 definitions and notations are fixed. In Section 3 related papers are mentioned and Elias’ sphere argument is explained, as well as the improvements made of it. In Section 4 the theoretical results of this paper may be found. In Section 5 examples of codes over asymmetric PSK which are better than their counterparts on symmetric PSK are presented, as well as proof that several classes of codes are optimal among codes over symmetric PSK. Section 6 concludes the paper with conclusions and possible continuations.

2. Definitions and notation

Some of the definitions below are more general than what is needed to obtain the results of this paper. To give a description of Elias’ sphere argument which is as general as possible, general definitions are necessary.

**Definition 2.1** Forney (1991): A space is said to be geometrically uniform (GU) if the distances from any element to all other elements are all the same, and all Voronoi regions have the same shape.

We will use $X$ to denote an arbitrary space. To denote a finite GU space we will use $U$. We will let $T$ denote the space of points on the unit circle, represented by their angle. We identify an angle $\alpha$ with all angles $\alpha + n2\pi$, $n \in \mathbb{Z}$. Whenever we speak of the sign of an angle, we consider the sign of its representative in the interval $(-\pi, \pi]$. The sign function, $\theta/|\theta|$ for $\theta \neq 0$, and 0 otherwise, will be denoted $\langle \theta \rangle$. We will use $Q$ to denote a finite GU subspace of $T$, and denote the $q = |Q|$ elements of $Q$ by their angle. We name those symbols $\gamma_0, \ldots, \gamma_{q-1}$, and by an argument of rotation, we will assume $\gamma_0 = 0$.

**Definition 2.2** Kschischang et al. (1989): Let $X$ be a space. A distance measure is a bivariate function $d$ from $X \times X$ to $\mathbb{R}$ such that

- $d(x, y) \geq 0$ for all $x, y \in X$ with equality if and only if $x = y$ and
- $d(x, y) = d(y, x)$ for all $x, y \in X$.

A distance measure which satisfies the triangle inequality, i.e. $d(x, z) \leq d(x, y) + d(y, z)$ for all $x, y, z \in X$ is a metric.

The distance measure $d$ will be said to be additive if

$$d(x, y) = \sum_{i=1}^{m} d(x_i, y_i)$$

for all $x, y \in X^m$ for all $m$.

The distance measure $d$ will be said to be translation-invariant if

$$d(x, y) = d(x + z, y + z)$$

for all $x, y, z \in X^m$ for all $m$.

Note that by saying that a distance measure is additive, one extends it from $X \times X$ to $X^m \times X^m$ for all $m$. 

---

Paper I: PSK Block Codes
For any distance measure $d$ we will denote $d(0, x)$ by $d(x)$. Also, we will refer to $d(x)$ as the weight of $x$ with respect to $d$.

**Definition 2.3** We define a family of additive and translation-invariant distance measures $\{d_p\}_{p>0}$ on $T$ or $Q$ by

$$d_p(\gamma) = 2^p \left| \sin \left( \frac{\gamma}{2} \right) \right|^p.$$  

Note that using phase-modulated sine-waves for communication and performing ML decoding while experiencing white noise $X$ with density

$$f_X(x) = \frac{\lambda^p}{2\Gamma(p-1)} \exp\left(-\frac{\lambda^p}{2}|x|^p\right)$$

for $p, \lambda > 0$, a correct way of measuring distance between symbols as well as words is by $d_p$. The density function (2.2) was introduced by Subbotin, Subbotin (1923), who suggested it to be used for modeling measurement errors. Subbotin found this density from the two properties:

1. The probability of an error depends only on the magnitude of this error, and may be expressed as a function $f_X(x)$ having the first derivative in general continuous.

2. The most probable value of a quantity, from which direct measures are known, must not depend on the unit employed.

Regarding the differentiability of $f_X(x)$ -- Subbotin’s derivation of $f_X(x)$ appears to rely on $f_X(x)$ being everywhere right differentiable (or left differentiable), but not on the derivative of $f_X(x)$ being continuous. The density in (2.2) is called generalized $p$-Gaussian density, and has been analyzed further in for example Pham & DeFuiereido (1989); Taguchi (1978).

**Notation 2.3** In the special case $p = 2$, the noise described in (2.2) is AWGN, and $d_2$ is commonly known as squared Euclidean distance. We signify the importance of this distance measure by denoting it $d_2^E$. Note that $d_2^E$ is not a metric.

Euclidean distance, $\sqrt{d_2^E}$, which will be denoted $d_E$, is a translation-invariant metric, but it is not additive. We will use that

$$d_E(\gamma) = 2 \left( \frac{\gamma}{2} \right) \sin \left( \frac{\gamma}{2} \right).$$

Remark that $\gamma/2$ is not well defined as it has two possible values, one positive and one negative. To get unique division in $T$, we lift the angle -- i.e. its representative in $(-\pi, \pi]$ -- to $\mathbb{R}$, perform the division, and bring the result back into $T$. For division by 2, this means that among the two possible values, we always choose the one which maintains the sign.

**Definition 2.4** The minimum distance between points in a space $X$ with respect to the distance measure $d$ is denoted as $d_{\min}(X)$ and defined as

$$d_{\min}(X) = \min_{x, y \in X, x \neq y} d(x, y).$$

The average distance between points in a space $X$ with respect to the distance measure $d$ is denoted as $d_{\text{average}}(X)$ and defined as infinity if $|X| \leq 1$ and as

$$d_{\text{average}}(X) = \frac{1}{|X|(|X|-1)} \sum_{x, y \in X, x \neq y} d(x, y)$$

otherwise.
Notation 2.4 A set of consecutive integers \( \{i, i+1, \ldots, j\} \) will often be denoted by \([i, j]\).

Notation 2.5 By \( \mathbb{Z}_+ \) we refer to the set of positive integers.

3. Previous results

In this paper we work with GU signal constellations. For a comprehensive text on GU codes and signal sets, see Forney (1991).

Elias did not publish his sphere argument, but the method may be found in Berlekamp (1968) pp. 318–321 and MacWilliams & Sloane (1977) pp. 558–564. We give an overview of Elias’ sphere argument for forming a bound on performance for codes over GU spaces together with generalizations.

Consider the code \( C \) on the GU space \( U^n \), i.e. let \( C \subseteq U^n \) be a code. Let \( d \) and \( \delta \) be two additive and translation-invariant distance measures, where we seek \( d_{\min}(C) \). Let

\[
S_{\delta,t}(x) = \{ y \in U^n | \delta(x,y) \leq t \}
\]

be a sphere with respect to \( \delta \), with center \( x \) and radius \( t \). We note that since we have a GU space and a translation-invariant distance measure, the number of words in the sphere \( S_{\delta,t}(x) \) is independent on \( x \).

Now, centered at each codeword, place a sphere with respect to \( \delta \) and radius \( t \). The spheres contain a total of \(|C||S_{\delta,t}|\) words, but only \(|U|^n\) words exist. So at least one word, \( w \), belong to at least \( K = \lceil |C||S_{\delta,t}|/|U|^n \rceil \) spheres. Call the sphere \( S_{\delta,t}(w) \) the critical sphere. Note that the critical sphere contains at least \( K \) codewords. Let \( W = C \cap S_{\delta,t}(w) \). Now, the chain of inequalities

\[
d_{\min}(C) \leq d_{\min}(W) \leq d_{\text{average}}(W)
\]

is trivial, so \( d_{\text{average}}(W) \) will work as an upper bound on \( d_{\min}(C) \).

Since we have a GU signal constellation it may as well be assumed that \( w = 0 \), as the code can be translated without changing any distances. With the assumption \( w = 0 \) we may speak of the weight with respect to \( \delta \) of codewords in the critical sphere, rather than of their distance to \( w \), and this weight is at most \( t \).

Elias defined \( \delta \) to be the distance measure in which the performance of the code was measured, i.e. \( \delta = d \), but this is not necessary. Improvements that have been made in Nilsson & Lennerstad (2000, 2005); Nilsson et al. (2008); Laksman et al. (2010, 2009a,b) are based on that while there is one distance measure (which we call outer distance measure) used for measuring the minimum distance of the code, another distance measure (which we call inner distance measure) can be used for determining the shape of the spheres, i.e. the one called \( \delta \). For PSK codes with AWGN it is squared Euclidean distance which is the relevant outer distance measure. The signal constellations symmetric 2-PSK, 3-PSK and 4-PSK are special cases, where squared Euclidean distance can be reduced to Hamming distance. The inner distance measure ought to be chosen such that it minimizes the bound in the outer distance measure.

Based on Elias’ sphere argument, but by allowing different inner distance measures, in Laksman et al. (2010) the following Theorem and Lemma were proven.

**Theorem 3.1** For any code \( C \) in \( Q^n \) we have the bound

\[
d_{\min}(C) \leq \min_{K \in [2,|C|]} \min_{\delta} \min_{\hat{y}} \frac{2I_K f_\delta(\hat{y})}{K-1},
\]
where

\[ \tilde{t}_K(\delta) = \min\{t : K \leq \lceil |C||S_{\delta,t}|^q n \rceil \}, \]

\[ f_\delta(y) = \frac{\sum_{j=1}^{K-1} \sum_{j=1}^{j-1} d(y_{j_1}, y_{j_2})}{\sum_{j=1}^{K} \delta(y_j)}, \]

and

\[ \tilde{y} = \arg\max_{y \in Q^K} f_\delta(y). \]

Remark that \( y \) in the theorem has length \( K \), not \( n \). The vector \( y \) is not a codeword, but a “column”, stemming from the construction in Berlekamp (1968) of forming a matrix of all codewords in a neighbourhood of radius \( \tilde{t}_K \), where the codewords are rows, so that column \( i \) consists of the \( i \)th symbols from the codewords in the critical sphere. Also, even though \( \tilde{t}_K \) is a function not only of \( \delta \), but also of \( n \), \( |C| \) and \( Q \), we usually omit those parameters as they are assumed to be fixed. The same is true for the dependence \( f_\delta(\tilde{y}) \) has on \( Q \). We also remark that the minimum over \( t \) always exists since the sphere \( S_{\delta,t} \) is defined with an inclusive inequality.

Lemma 3.1 The bound in Theorem 3.1 is scale invariant in the distance measure \( \delta \), i.e. for any \( s > 0 \), let \( \lambda(x) = s\delta(x) \) for every \( x \). Then

\[ \frac{2\tilde{t}_K(\delta) f_\delta(\tilde{y})}{K-1} = \frac{2\tilde{t}_K(\lambda) f_\lambda(\tilde{y})}{K-1} \]

holds.

While both Theorem 3.1 and Lemma 3.1 in Laksman et al. (2010) were considered only for symmetric PSK and for squared Euclidean distance \( d \), proofs for asymmetric PSK and general additive translation-invariant \( d \) remain nearly identical. Note however that Theorem 3.1 does not in general hold if the PSK signal set is not GU.

In Nilsson & Lennerstad (2000, 2005); Nilsson et al. (2008); Laksman et al. (2010, 2009b,a) the idea of using an inner distance measure different from the outer distance measure has been developed. Note that while some partial results from these papers may hold for a general outer distance measure, they were all developed with the restriction to squared Euclidean distance as outer distance measure. In Nilsson & Lennerstad (2000, 2005) particular inner distance measures which improved the Elias’ bound for some code parameters were presented, while in Nilsson et al. (2008) the idea of letting the inner distance measure depend on \( K \) appeared, using symmetric 8-PSK to exemplify. How \( \delta \) ought to depend on \( K \) for symmetric 8-PSK was studied in Laksman et al. (2010), which in Laksman et al. (2009b) was generalized to symmetric \( q \)-PSK. In Laksman et al. (2009a) it was shown that the work in Laksman et al. (2010), somewhat limited by the method, gave the actual optimum, but this was only for \( q = 8 \), and failed to extend to general \( q \). This extension is provided in the present paper, besides the generalizations to general outer distance measures and asymmetric PSK.

4. Results

Generalizations built upon Elias’ sphere argument have earlier been made for block codes over symmetric PSK with squared Euclidean distance, Nilsson & Lennerstad (2000, 2005); Nilsson et al. (2008); Laksman et al. (2010, 2009b,a). The main methods in this section are further generalizations built on the generalizations previously made. In the present paper the earlier generalizations are shown to work for
block codes over asymmetric PSK. Furthermore, it is shown that any additive and translation-invariant outer distance measure can be used. The end of the section however, deals explicitly with the case where the outer distance measure is squared Euclidean distance.

Let \( a_i(y) \) be the function counting the number of symbols equal to \( \gamma_i \) in a vector \( y \). Now write \( f_\delta \) from Theorem 3.1 as

\[
f_\delta = \frac{\sum_{i=0}^{q-1} \sum_{j>i} a_i a_j d(\gamma_i, \gamma_j)}{\sum_{i=0}^{q-1} a_i \delta(\gamma_i)}.
\]

Rewriting the expression, using \( a_0 = K - \sum_{i=1}^{q-1} a_i \) as a way of eliminating \( a_0 \), we find that numerator has the following coefficients

\[
\begin{align*}
\text{coeff}(a_i) &= K d(\gamma_i), \\
\text{coeff}(a_i^2) &= -d(\gamma_i) \quad \text{and} \\
\text{coeff}(a_i a_j) &= d(\gamma_i, \gamma_j) - (d(\gamma_i) + d(\gamma_j)), \quad i \neq j.
\end{align*}
\]

Now we may write

\[
f_\delta = \frac{T_1 + T_2 + T_3}{D}, \quad \text{with}
\]

\[
T_1 = K \sum_{i=1}^{q-1} a_i d(\gamma_i), \\
T_2 = -\sum_{i=1}^{q-1} a_i^2 d(\gamma_i), \\
T_3 = \sum_{i=1}^{q-1} \sum_{j>i} a_i a_j (d(\gamma_i, \gamma_j) - (d(\gamma_i) + d(\gamma_j))),
\]

\[
D = \sum_{i=1}^{q-1} a_i \delta(\gamma_i).
\]

**DEFINITION 4.1** Columns \( y \) which maximize \( f_\delta(y) \) for at least some \( \delta \) will be called **extremal columns**. A column \( y \) which maximizes \( f_\delta(y) \) for a particular \( \delta \) will be called a **critical column** for that \( \delta \).

Note that which columns are extremal, and which column is critical for a particular \( \delta \), depends on the outer distance measure, which we however assume to be fixed.

**THEOREM 4.1** Let \( y \) be a column such that \( T_3(y) < 0 \), then \( y \) is not an extremal column.

**Proof.** Let \( Q_\delta \) be the set of symbols in \( y \) except \( \gamma_0 \). Let

\[
m = \arg \max_{\{i : \gamma_i \in Q_\delta\}} \left( \frac{d(\gamma_i)}{\delta(\gamma_i)} \right).
\]

Let \( z \) be a column with \( a_0(z) = K - 1 \) and \( a_m(z) = 1 \). As \( T_3(z) = 0 \) it follows that

\[
f_\delta(z) = \frac{T_1 + T_2}{D}(z) = \frac{(K - 1) d(\gamma_m)}{\delta(\gamma_m)}.
\]
As \( T_3(y) < 0 \), it follows that

\[
\frac{f_\delta(y)}{D(y)} = \frac{\sum_{i=1}^{q-1} a_i(y)(K - a_i(y))d(\gamma_i)}{\sum_{i=1}^{q-1} a_i(y)d(\gamma_i)} \leq \frac{\sum_{i=1}^{q-1} a_i(y)(K - 1)d(\gamma_i)}{\sum_{i=1}^{q-1} a_i(y)d(\gamma_i)} \leq \frac{(K - 1)d(\gamma_m)}{\delta(\gamma_m)}.
\]

The last inequality follows from the left-hand side being a weighted average of several terms, where the right-hand side is the largest of those terms. In total, we have shown that \( f_\delta(y) < f_\delta(x) \), so \( y \) cannot be an extremal column.

Finding the optimal inner distance measure is simplified a great deal when the outer distance measure is a metric, as shown by a corollary to Theorem 4.1.

**COROLLARY 4.1** If the outer distance measure \( d \) is a metric, then an optimal inner distance measure \( \delta \) is the same as the outer distance measure.

**Proof.** If \( d \) is a metric, then \( T_3 \leq 0 \). Following the proof of Theorem 4.1, we see that columns of the form \((i, 0, \ldots, 0)^T\) for all \( i \) are extremal, and for any \( \delta \) one of these columns will be critical, even if other extremal columns which may be simultaneously critical may exist. Since interest lies in the maximal value of \( f \), and not in extremal columns as such, the set of columns \((i, 0, \ldots, 0)^T\) will be sufficient.

For any given \( \delta \), one of these columns will be \( \hat{y} \) in Theorem 3.1, so the bound is

\[
\min_{k, \delta} \frac{2T_k(\delta)}{K - 1} \max_i \left( \frac{(K - 1)d(\gamma_i)}{\delta(\gamma_i)} \right) = \min_{k, \delta} \frac{2T_k(\delta)}{\delta(\gamma_i)} \max_i \left( \frac{d(\gamma_i)}{\delta(\gamma_i)} \right).
\]

Consider an inner distance measure \( \delta_0 \) with

\[
m = \arg\max_i \left( \frac{d(\gamma_i)}{\delta_i(\gamma_i)} \right).
\]

Lemma 3.1 allows us to rescale \( \delta_0 \) so that \( \delta_0(\gamma_m) = d(\gamma_m) \). If \( \delta_0(\gamma_i) > d(\gamma_i) \) for some \( i \), consider \( \delta_1 \) such that \( \delta_1(\gamma_i) = d(\gamma_i) \) and \( \delta_1(\gamma_j) = \delta_0(\gamma_j) \) for \( j \neq i \). Clearly the bound resulting from using \( \delta_1 \) as inner distance measure does not give a worse bound than using \( \delta_0 \) as inner distance measure. The process may be repeated until we have an inner distance measure \( \delta = d \).

When an optimal inner distance measure is known, Theorem 3.1 can be applied directly.

The last corollary applies to e.g. Hamming metric

\[
d_H(\gamma_i, \gamma_j) = \begin{cases} 0, & \text{if } \gamma_i = \gamma_j \\ 1, & \text{if } \gamma_i \neq \gamma_j \end{cases}
\]

and to Lee metric

\[
d_L(\gamma_i, \gamma_j) = \arcsin(d_1(\gamma_i, \gamma_j)/2),
\]

both of which are commonly used for determining performance of error-correcting codes. However, for these distance measures other upper bounds are known. We recommend van Lint (1992); Roth (2006) for the Hamming metric and Quistorff (2006) for the Lee metric.

Note that it is enough for \( d \) to be a metric on \( Q \) for Corollary 4.1 to hold, there is no need for \( d \) to be a metric on the whole interval \((-\pi, \pi]\). For example, squared Euclidean distance is a metric for symmetric 4-PSK, but not in general. Also, for \( p \in (0, 1] \), \( d_p \) (from (2.1)) is a metric on the whole interval \((-\pi, \pi]\). It is possible to formulate a relation between the sizes of \( p \) which yield a metric and the angle of the two symbols closest to zero.
THEOREM 4.2 For $p > 1$, the distance measure $d_p$ from (2.1) is a metric on an asymmetric PSK signal-set $Q$ if and only if
\[
\sin \left( \frac{\gamma_i - \gamma_{q-1}}{2} \right)^p \leq \sin \left( \frac{\gamma_i}{2} \right)^p + \sin \left( \frac{\gamma_{q-1}}{2} \right)^p,
\]
where $\gamma_i$ and $\gamma_{q-1}$ are the two smallest angles of $Q$, see Figure 1(b).

Proof. Note that (4.2) is in fact the triangle inequality for 3 particular elements in $Q$. If we identify it with Definition 2.2, we have $x = \gamma_1, y = \gamma_0, z = \gamma_{q-1}$. By a rotational argument, we can choose $y = \gamma_0$, no matter which 3 elements we intend to “test” the triangle inequality on.

Based on the observation that (4.2) is an instance of the triangle inequality, it is clear that $d_p$ is not a metric if (4.2) is false.

For the other direction, we first note that if $x$ and $z$ have the same sign (note that $|x|, |z| \leq \pi$), then clearly
\[
\sin \left( \frac{x-z}{2} \right)^p \leq \sin \left( \frac{x}{2} \right)^p + \sin \left( \frac{z}{2} \right)^p.
\]
Hence, it is enough to show that (4.3) holds for all $x \in [\gamma_1, \pi]$ and $z \in (-\pi, \gamma_{q-1}]$. With $x$ and $z$ in those intervals, (4.3) can be written as
\[
0 \leq \sin^p \left( \frac{x}{2} \right) + \sin^p \left( \frac{-z}{2} \right) - \sin^p \left( \frac{x-z}{2} \right).
\]
It is an easy analytic exercise to show that the right-hand side is increasing with respect to $x$ and decreasing with respect to $z$, for $x$ and $z$ in the given intervals, so knowing that the inequality is true for the minimum $x$ and the maximum $z$ is enough. Hence, $d_p$ must be a metric on $Q$ when (4.2) is true.

When the signal constellation comes from symmetric PSK, (4.2) can be rewritten in a much cleaner form.

COROLLARY 4.2 The distance measure $d_p$ from (2.1) is a metric on the symmetric PSK constellation $Q$ if and only if
\[
p \leq \left( \log_2 \left( \frac{\sin (\gamma_1)}{\sin (\gamma_1/2)} \right) \right)^{-1},
\]
where $\gamma_1$ is the smallest angle of $Q$, see Figure 1(a).

Proof. The corollary is a matter of rewriting (4.2) with $\gamma_i = -\gamma_{q-1}$. For the next theorem – which helps in reducing the set of possibly extremal columns – denoting
\[
a_i(x) = (a_1(x), \ldots, a_{q-1}(x))
\]
and
\[
N(x) = f_\delta(x) \delta(x) = \sum_{i=0}^{q-1} \sum_{j>i} a_i(x) a_j(x) d(\gamma_i, \gamma_j)
\]
will be helpful. The following theorem was presented in a less general setting already in Laksman et al. (2010).

THEOREM 4.3 If a column $z$ is extremal, then
\[
\frac{N(z)}{\delta(z)} \geq \frac{\sum_{i=1}^{m} u_i N(y_i)}{\sum_{i=1}^{m} u_i \delta(y_i)}
\]
for all \(y_1, y_2, \ldots, y_m\), for all \(m\), such that for at least some \(v, u_i \in \mathbb{Z}_+\),
\[
\sum_{i=1}^{m} u_i a_i(y_i) = v a_+(z)
\]
for all finite \(\delta\).

Note that the denominator will be the same on both sides in (4.4), at least once the left-hand side has been extended by \(v\), so \(\delta\) will play no part in the evaluation of the inequality. Also the different \(T_i\)-terms will cancel, so the parameter \(K\) will disappear, and affect (4.4) only by limiting the number of symbols different from \(\gamma_0\) in each column, so it is only a comparison of constants. 

**Proof.** Assume that for some \(m\), a set of columns \(y'_1, y'_2, \ldots, y'_m\) exist such that for some \(v, u_i \in \mathbb{Z}_+, \sum_{i=1}^{m} u_i a_i(y'_i) = v a_+(z)\) and \(N(x) \sum_i u_i d(y'_i) = \delta(x) \sum_i u_i N(y'_i) < 0\). Then for any \(\delta\), we must have \(N(x) \delta(y'_i) - \delta(x) N(y'_i) < 0\) for at least some \(i\), which is the same as saying that \(z\) is not extremal.

Let \(\tilde{y}_1, \ldots, \tilde{y}_{c(K)}\) be a list containing all extremal columns. It may also contain other columns, but for computational purposes it is preferable if it does not. Note that this list may depend somewhat on \(K\), as the length of the columns is limited by \(K\).

To be able to truly utilize the previous theorem, it would be good to have a set of extremal columns to use for the further reduction. Let \(\gamma_i\) be a symbol in \(Q\). We consider two cases, depending on whether \(\gamma_i\) has an additive inverse in \(Q\) or not.

**Case 1, \(-\gamma_i \notin Q\):**
Consider now the inner distance measure \(\delta\) defined by
\[
\delta(x) = \begin{cases} 
0, & x = \gamma_0, \\
1, & x = \gamma_i, \\
\infty, & \text{otherwise}
\end{cases}
\]

This \(\delta\) shows that an extremal column \(x\) with \(a_j(x) = 0\) for all \(j \notin \{0, i\}\) exists. With this condition on \(x\), it can be seen that \(f_\delta\) is maximized if \(a_i(x) = 1\), as \(T_1(x)/D(x)\) is constant with respect to \(a_i(x)\). \(T_2(x)/D(x)\) is decreasing with respect to \(a_i(x)\) and \(T_3(x) = 0\). Thus if \(-\gamma_i \notin Q\), then \((\gamma_i, 0, \ldots, 0)^T\) is an extremal column.

**Case 2, \(-\gamma_i \in Q\):**
Consider now the inner distance measure \(\delta\) defined by
\[
\delta(x) = \begin{cases} 
0, & x = \gamma_0, \\
1, & x = \pm \gamma_i, \\
\infty, & \text{otherwise}
\end{cases}
\]  

(4.5)

This \(\delta\) shows that an extremal column \(x\) with \(a_k(x) = 0\) for all \(k \notin \{0, i, j\}\) exists. With this condition on \(x\), \(f_\delta\) must be maximized with respect to \(a_i\) and \(a_j\). As \(T_1/D\) is constant with respect to \(a_i\) and \(a_j\), the value of \(T_1\) causes no concern. Instead we observe that \(a_i^2 + a_j^2 \geq 2a_i a_j\), with equality when \(a_i = a_j\), so that
\[
T_2 + T_3 \leq a_i a_j (d(\gamma_i, \gamma_j) - 3(d(\gamma_i) + d(\gamma_j))) = a_i a_j (d(2\gamma_i) - 6d(\gamma_i)).
\]

This means that if \(d(2\gamma_i) > 6d(\gamma_i)\), then a column maximizing \(f_\delta\) yield \(a_i + a_j \in \{K - 1, K\}\) and \(|a_i - a_j| \leq 1\). Otherwise, \(a_i\) and \(a_j\) should be kept as small as possible, so that \(a_i + a_j \in \{1, 2\}\) and \(|a_i - a_j| \leq 1\).
Case 2 above indicates that for certain distance measures some extremal columns will contain zeroes in all but at most two positions, whereas for other distance measures extremal columns consisting of at most one zero exist.

Now follows a theorem which allows the optimization of $\delta$, which is non-enumerable, to be calculated as a finite number of linear programming problems (LPPs). The idea of this theorem was used in Laksman et al. (2009a), albeit in a far more restricted setting.

**THEOREM 4.4** The value of

$$\min_{\delta \in \Delta} \frac{2}{K-1} f_{K}(\delta) \max_{i \in [1, c(K)]} f_{\delta}(\tilde{y}_i)$$  \hspace{1cm} (4.6)

where $K \geq 2$ is an integer and $\Delta$ is the set of additive translation-invariant distance measures, is the same as the value of

$$\min_{i \in [1, c(K)]} \min_{\delta \in \Delta} \frac{2}{K-1} f_{K}(\delta) f_{\delta}(\tilde{y}_i)$$  \hspace{1cm} (4.7)

where $\Delta_i$ is the set of additive translation-invariant distance measures which solves the linear programming problem

$$\begin{align*}
\max_{\delta} & \quad \delta(\tilde{y}_i) \\
\text{s.t.} & \quad \sum_{j=0}^{q-1} \delta(\gamma_j) = A_i \\
& \quad N(\tilde{y}_i) \delta(\tilde{y}_j) - N(\tilde{y}_j) \delta(\tilde{y}_i) \geq 0, j \neq i,
\end{align*}$$

where $A_i$ is any positive constant.

**Proof.** For now, assume that for the $\delta$ which is the minimizing argument in (4.6), a critical column, i.e. a column $\tilde{y}_1$ which maximizes $f_{\delta}$, is $\tilde{y}_1$.

We have

$$f_{\delta}(\tilde{y}_i) = \frac{N(\tilde{y}_i)}{\delta(\tilde{y}_i)}.$$

By the lemma of scale invariance, Lemma 3.1, we see that to perform the $\delta$-minimization in (4.6) we should maximize $\delta(\tilde{y}_i)$ subject to the constraints

$$f_{\delta}(\tilde{y}_1) \geq f_{\delta}(\tilde{y}_i), \quad \forall i \in [2, c(K)],$$

which may also be expressed as

$$N(\tilde{y}_1) \delta(\tilde{y}_i) - N(\tilde{y}_i) \delta(\tilde{y}_1) \geq 0, \quad \forall i \in [2, c(K)].$$

But to maximize $\delta(\tilde{y}_i)$ under the conditions (4.8) is just a linear programming problem.

We assumed that $\tilde{y}_1$ was a critical column, but we have to solve the LPP for any column being critical, so to optimize the bound for a given $K$ we must solve $c(K)$ LPPs. For any fixed critical column, we may choose the distance measure resulting in the lowest bound. To get the best bound possible, we must try for each extremal column being critical, and pick the smallest one. This is expressed by the minimization over $i$.

The condition $\sum_j \delta(\gamma_j) = A_i$ is just to fix the scaling, and can be added without loss of generality, as the problem we wish to solve is invariant with respect to scaling. Without this condition, the feasible
Note that we get no real problem from the LPPs where columns $\hat{y}_i$ which are not extremal are included in our list, because the feasible region for such columns will be empty.}

Note that while the outer distance measure $d$ does not appear explicitly in either Theorem 4.3 or 4.4, it appears implicitly in both $N(x)$ and in $f_\delta(x)$, so both theorems are generalized with respect to the shape of the noise.

Now a theorem reducing the bound from Theorem 3.1 is presented. The set of words is restricted to those that are close to the radius of the sphere only, introducing a benevolent cancellation in at least one case. For the theorem we denote

$$B_K(\delta) = \frac{2\pi_k(\delta) \max_y f_\delta(\hat{y})}{K-1},$$

and observe that for any $q, n, |C|$ and any $K$ and $\delta$, $B_K(\delta)$ is a bound on the maximal minimum squared Euclidean distance for a code with parameters $q, n$ and $|C|$.

This improvement exploits cancellation in the differences and the fact that the bound is not dependent on the block length.

**Theorem 4.5** We have the bound

$$d_{\min} \leq \max_{(x,y) \in R} d(x,y),$$

where

$$R = \{(r_1, r_2) \in Q^n \times Q^n : \hat{r}_k(\delta) - \delta(\gamma_i) < \delta(r_i) \leq \hat{r}_k(\delta), i = 1, 2 \text{ and } d(r_1, r_2) \leq B_K(\delta)\},$$

for $\gamma_n$ such that $\delta(\gamma_n) = \min_{i \neq 0} \delta(\gamma_i)$.

The theorem is an extension of Theorem II.2 in Nilsson & Lennerstad (2000), where the quantity $ad_E^2(\pi/2) + bd_E^2(\pi/4)$ was to be maximized constrained by the bound of the paper, similarly to Theorem 4.5. The reformulation changed the quantity to $ad_E^2(\pi/2) + 2bd_E^2(\pi/4)$, introducing a factor 2 in the second term. The basic idea is to demand that minimum distance can be written as the distance between two codewords being sufficiently close to the edge of the critical sphere.

Before proving Theorem 4.5, an example of how it can be used is given.

**Example 4.6** Consider the signal constellation of symmetric 8-PSK and $d = d_E^2$. Let $t$ and $\delta$ be such that $S_{\delta,t}(\emptyset)$ contains only words consisting of zeros and at most two $\pm \pi/4$'s. Obviously $\delta(\pi/4) = \min_{\gamma \neq 0} \delta(\gamma)$, otherwise some other words would be included in the sphere. Also, $t$ must be at least $2\delta(\pi/4)$, implying that the set $R$ described in Theorem 4.5 contains only all words with exactly two $\pm \pi/4$'s and all other symbols zero. The only differences which can appear between two words in $R$ are $2d_E^2(\pi/2), d_E^2(\pi/2) + 2d_E^2(\pi/4), 4d_E^2(\pi/4), d_E^2(\pi/2), 2d_E^2(\pi/4)$ and 0.

Assuming

$$4d_E^2(\pi/4) < d_E^2(\pi/2) + d_E^2(\pi/4) \leq B_K(\delta) < d_E^2(\pi/2) + 2d_E^2(\pi/4),$$

we could conclude that the bound is as low as $4d_E^2(\pi/4)$.

It is not guaranteed that the best bound is obtained by first minimizing $B_K(\delta)$ with respect to $K$ and $\delta$, as some other choice of $K$ and $\delta$ may allow for better use of Theorem 4.5, but it is a reasonable way to proceed in order to achieve a good bound.
Proof. A sequence of symbols $x$ is in this proof called maximal if $\tilde{t}_K(\delta) - \delta(\gamma_m) < \delta(x) \leq \tilde{t}_K(\delta)$, non-maximal if $\delta(x) \leq \tilde{t}_K(\delta) - \delta(\gamma_m)$, and not allowed if $\tilde{t}_K(\delta) < \delta(x)$. This theorem is possible since the bound is independent on the block length, since the function $f_\delta$ in Theorem 3.1 represents the worst column in a matrix $M_W$ where the rows are the words in $W$. So we relax the constraint that a sequence of symbols is limited to having $n$ symbols. Instead we consider possibly infinite sequences. For each non-maximal sequence in $W$ we add one column to the matrix with a non-zero symbol in the row of the non-maximal sequence, so the sequence is still allowed. This is possible, since at least $\delta(\gamma_m)$ always can be added to a non-maximal sequence. All other entries in the column are zero. In the same way columns are added to the matrix until all sequences are maximal, whereafter we may continue to add as many columns with only zeros as we like. This gives in the end a set $W'$ and matrix $M_{W'}$. Clearly, $d_{\text{min}}(W) \leq d_{\text{min}}(W')$, as the distance between two sequences in $W'$ are never smaller than the distance between the corresponding words in $W$. The theorem considers the maximum of all differences of maximal sequences fulfilling the bound $B_K(\delta)$, which bounds $d_{\text{min}}(W')$ from above. The result follows.

As a common assumption is that one operates under AWGN, it is reasonable to pay some additional attention to the case with $d_E^2$ as outer distance measure. It turns out that a useful reformulation of $f_\delta$ is possible when $d = d_E^2$. In particular, this reformulation permits a theorem for reducing the set of extremal columns, not completely unlike that of Theorem 4.1.

We consider the expression for $f_\delta$ given in (4.1) and simplify it by using $d = d_E^2$, which allows writing $-(T_2 + T_3)$ as a sum of two squares. Note that as

$$2(\sin^2(\theta - \vartheta) - \sin^2(\theta) - \sin^2(\vartheta)) = -4\sin^2(\theta)\sin^2(\vartheta) - \sin(2\theta)\sin(2\vartheta), \quad (4.9)$$

for any $\theta$ and $\vartheta$, we get

$$4T_3 = -2\sum_{i=1}^{q-1} \sum_{j>i} a_i a_j (d_E^2(\gamma_i)d_E^2(\gamma_j) + (2\gamma_i/2)(2\gamma_j/2)) d_E(2\gamma_i)d_E(2\gamma_j) =$$

$$= -2\sum_{i=1}^{q-1} \sum_{j>i} a_i a_j (d_E^2(\gamma_i)d_E^2(\gamma_j) + \gamma_i \gamma_j d_E(2\gamma_i)d_E(2\gamma_j)).$$

Note that we may cancel the 2’s in the expression $\langle 2(\theta/2) \rangle$ only because the division comes first (the operations are not associative in $T$). That division comes first follows from that the division is implicit already in the first equation describing $T_3$.

As a special case of (4.9), using $\theta = \vartheta$, the following equality

$$-4\sin^2(\theta) = -4\sin^4(\theta) - \sin^2(2\theta)$$

is obtained, and yields

$$4T_2 = -\sum_{i=1}^{q-1} a_i^2 (d_E^2(\gamma_i) + d_E^2(2\gamma_i)).$$
Now observe that

\[
- \left( \sum_{i=1}^{q-1} a_i d_E^2(\gamma_i) \right)^2 - \left( \sum_{i=1}^{q-1} a_i \langle \gamma_i \rangle d_E(2\gamma_i) \right)^2 = 4T_2 - 2 \sum_{i=1}^{q-1} \sum_{j>i} a_i a_j (d_E^2(\gamma_i) d_E^2(\gamma_j) + \langle \gamma_i \rangle \langle \gamma_j \rangle d_E(2\gamma_i) d_E(2\gamma_j)) = 4(T_2 + T_3).
\]

Thus, we may write

\[
f_\delta = \frac{4T_1 - \left( \sum_{i=1}^{q-1} a_i d_E^2(\gamma_i) \right)^2 - \left( \sum_{i=1}^{q-1} a_i \langle \gamma_i \rangle d_E(2\gamma_i) \right)^2}{4D}.
\]

Theorem 4.1 and the following theorem may appear to be similar, but they are independent:

**Theorem 4.7** The condition

\[
\sum_{i=1}^{q-1} a_i(x) d_E^2(\gamma_i) = d_E^2(x) \leq 4
\]

holds for any extremal column \(x\).

The theorem is a generalization of a result from Laksman *et al.* (2009b). *Proof.* Assume that \(x\) is an extremal column such that

\[
\sum_{i=1}^{q-1} a_i(x) d_E^2(\gamma_i) > 4,
\]

i.e., a \(\delta\) exists such that \(x = \text{argmax}_y f_\delta(y)\). For such a \(\delta\), let \(\gamma_m\) be a symbol such that

\[
\frac{d_E^2(\gamma_m)}{\delta(\gamma_m)} \geq \frac{d_E^2(\gamma_i)}{\delta(\gamma_i)} \quad \forall i \in [0, q-1]. \tag{4.10}
\]

Then we have

\[
\frac{d_E^2(\gamma_m)}{\delta(\gamma_m)} \geq \frac{\sum_{i=1}^{q-1} b_i d_E^2(\gamma_i)}{\sum_{i=1}^{q-1} b_i \delta(\gamma_i)}
\]

for any nonnegative constants \(b_1, \ldots, b_{q-1}\) whereof at least one is positive, as the right-hand side can be shown to be a weighted average of the quantities on the right-hand side of (4.10).

Now let \(y\) be a column such that \(a_m(y) = 1, a_0(y) = K - 1\), and thus \(a_i(y) = 0\) for all \(i \notin \{0, m\}\). Calculation shows that

\[
f_\delta(y) = (4K - 4) \frac{d_E^2(\gamma_m)}{4\delta(\gamma_m)}.
\]
Next we’ll prove that \( f_\delta(x) < f_\delta(y) \), so \( x \) cannot be extremal. Form a bound on \( f_\delta(x) \) as
\[
\frac{\left( \sum_{i=1}^{q-1} a_i(x) \gamma_i d_E(2\gamma_i) \right)^2}{4 \sum_{i=1}^{q-1} a_i(x) \delta(\gamma_i)} = 4K \frac{\sum_{i=1}^{q-1} a_i(x) d_E^2(\gamma_i)}{4 \sum_{i=1}^{q-1} a_i(x) \delta(\gamma_i)}
\]

But by assumption we have
\[
4K - 4 > 4K - \sum_{i=1}^{q-1} a_i(x) d_E^2(\gamma_i) \quad \text{and} \quad \frac{d_E^2(\gamma_m)}{4\delta(\gamma_m)} \geq \frac{\sum_{i=1}^{q-1} a_i(x) d_E^2(\gamma_i)}{4 \sum_{i=1}^{q-1} a_i(x) \delta(\gamma_i)},
\]

thus \( f_\delta(y) > f_\delta(x) \), which yields a contradiction. Hence, no extremal column \( x \) with \( d_E^2(x) > 4 \) can exist.

\[\square\]

5. Codes

For uncoded communication it is trivial to see that the symmetric version of PSK maximizes the minimum distance, but for coded communication it is not so. In Examples 5.1 and 5.2 block codes with asymmetric PSK which are better than optimal block codes with symmetric PSK are presented. The codes used in these examples are very small, and presented only to show that performance can be improved by use of asymmetric PSK rather than symmetric PSK. For practical use, larger codes need to be developed. As mentioned earlier, it has been shown that for certain code classes asymmetric PSK yields improvement over symmetric PSK, see Bali & Rajan (1998).

**Example 5.1** Consider a PSK code with \( q = 8, n = 2 \) and \( |C| = 3 \). The best such code given symmetric PSK and \( \gamma = 2\pi i/q \) is \( C = \{(\gamma_0, \gamma_0), (\gamma_2, \gamma_2), (\gamma_6, \gamma_6)\} \), which results in \( d_{E_{\min}}^2(C) \approx 5.41 \). The best such code given asymmetric PSK is obtained by using the same code, but with \( \gamma_i = 2\pi i/q \) only for even \( i \), and \( \gamma_i = 2\pi(i + 0.41)/q \) for odd \( i \), which results in \( d_{E_{\min}}^2(C) \approx 5.79 \).

The columns \( \hat{y}_i = (\gamma_i, \gamma_i, \ldots, \gamma_i)^T \), for \( i = 1, \ldots, 7 \) and \( \hat{y}_8 = (\gamma_1, \gamma_6, \gamma_0, \ldots, \gamma_0)^T \) are extremal. For \( K = 2 \), also \( \hat{y}_9 = (\gamma_1, \gamma_7)^T \) and \( \hat{y}_{10} = (\gamma_2, \gamma_7)^T \) are extremal. If \( K \geq 3 \), then \( \hat{y}_9 = (\gamma_1, \gamma_7, \gamma_0, \ldots, \gamma_0)^T \) and \( \hat{y}_{10} = (\gamma_2, \gamma_7, \gamma_0, \ldots, \gamma_0)^T \) are extremal.

With \( K = 2 \) and \( \delta(\hat{y}_8) \) critical we get
\[
\delta(\gamma_1) = 1.11, \quad \delta(\gamma_2) = 2.68,
\]
\[
\delta(\gamma_3) = 3.79, \quad \delta(\gamma_4) = 4.00,
\]
\[
\delta(\gamma_5) = 2.89, \quad \delta(\gamma_6) = 2.68,
\]
\[
\delta(\gamma_7) = 0.89,
\]
for \( A_8 = 18.05 \). This yields the bound 7.58, which can also be obtained by some other inner distance measures, and is the best bound attainable by this method.

That the code in the example is optimal within the set of codes over asymmetric PSK has been verified by exhaustive search.

36
Example 5.2 Consider a PSK code with $q = 8$, $n = 2$ and $|C| = 4$. The best such code given symmetric PSK and $\gamma_i = 2\pi i/q$ is $C = \{(\gamma_0, y_0), (\gamma_1, y_1), (\gamma_2, y_2), (\gamma_3, y_3)\}$, which results in $d_{E_{\text{min}}}^2(C) \approx 4.59$. The best such code given asymmetric PSK is obtained by using the same code, but with $\gamma_i = 2\pi i/q$ only for even $i$, and $\gamma_i = 2\pi(i + 0.57)/q$ for odd $i$, which results in $d_{E_{\text{min}}}^2(C) \approx 5.33$.

The columns $\tilde{y}_i = (\gamma_0, y_0, \ldots, y_0)^T$, for $i = 1, \ldots, 7$ and $\tilde{y}_8 = (\gamma_1, y_6, y_0, \ldots, y_0)^T$ are extremal. For $K = 2$, also $\tilde{y}_9 = (\gamma_1, y)^T$ and $\tilde{y}_{10} = (\gamma_2, y)^T$ are extremal. If $K = 3$, then $\tilde{y}_9 = (\gamma_1, y, y)^T$ and $\tilde{y}_{10} = (\gamma_2, y, y)^T$ are extremal. If $K \geq 4$, then $\tilde{y}_9 = (\gamma_1, y, y, y_0, \ldots, y_0)^T$ and $\tilde{y}_{10} = (\gamma_2, y, y, y_0, \ldots, y_0)^T$ are extremal.

With $K = 2$ and $\delta(\tilde{y}_8)$ critical we get

$$\delta(\gamma_1) = 1.70, \quad \delta(\gamma_2) = 2.18,$$
$$\delta(\gamma_3) = 3.89, \quad \delta(\gamma_4) = 4.00,$$
$$\delta(\gamma_5) = 2.66, \quad \delta(\gamma_6) = 2.18,$$
$$\delta(\gamma_7) = 0.48,$$

for $A_8 = 17.10$. This yields the bound 5.33 which is the best bound possible, though, many other inner distance measures result in the same bound.

That the code in this example is optimal within the set of codes over asymmetric PSK has been verified by exhaustive search. That the code is optimal within the set of codes over asymmetric PSK does not follow from its minimum distance meeting the bound, as this is only a bound for one particular asymmetric PSK constellation.

Examples 5.3 – 5.7, based on multilevel codes Imai & Hirakawa (1977), consider symmetric $q$-PSK and are intended to show how to apply the bound for large asymmetric PSK constellation. That the code in this example is optimal within the set of codes over asymmetric PSK has been verified by exhaustive search. That the code is optimal within the set of codes over asymmetric PSK does not follow from its minimum distance meeting the bound, as this is only a bound for one particular asymmetric PSK constellation.

Example 5.3 Consider $q = 3a$, where $a \geq 2$ is an integer. Then a code $C$ with $n = (3^m + 1)/2$, $|C| = a^{m-1}$ and $d_{E_{\text{min}}}^2(\gamma_i)$ exists, for any integer $m \geq 2$. It can be constructed as $C = C_1 + 3C_2$ where $C_1$ is a ternary extended Hamming code and $C_2$ is $Z_a^n$. The bound described in this work will be used to show that for large enough $m$ the code $C$ has the highest $d_{E_{\text{min}}}^2$ possible.

Consider the inner distance measure $\delta$ described in (4.5), with $i = 1$, and $r = 2\delta(\gamma_1)$. These $\delta$ and $t$ do not necessarily yield the lowest bound possible initially, but by use of Theorem 4.5 they will be sufficient. Theorem 4.5 gives that the bound must have the form (in order of size) $2d_E^2(\gamma_1)$, $d_E^2(\gamma_2)$, $4d_E^2(\gamma_1)$, $2d_E^2(\gamma_1) + d_E^2(\gamma_2)$ or $2d_E^2(\gamma_2)$.

The chosen $\delta$ results in $\tilde{y} = (\gamma_1, y_{q-1}, y_0, y_0, \ldots, y_0)^T$ and

$$f_\delta(\tilde{y}) = \frac{2Kd_E^2(\gamma_1) - d_E^2(\gamma_i)}{2\delta(\gamma_1)} = \frac{2(K - 2)d_E^2(\gamma_1) + d_E^2(\gamma_2)}{2\delta(\gamma_1)}.$$

The bound becomes

$$\frac{2f_\delta(\tilde{y})}{K - 1} = 2d_E^2 + \frac{2(K - 3)d_E^2(\gamma_1) + 2d_E^2(\gamma_2)}{K - 1}.$$

As $d_E^2(\gamma_2) > 2d_E^2(\gamma_1)$ for $q \geq 5$, we see that for $K \geq 4$ the bound is less than $2d_E^2(\gamma_1) + d_E^2(\gamma_2)$, which is enough to claim that it is $4d_E^2(\gamma_1)$. To show that $C$ is optimal with respect to $d_{E_{\text{min}}}^2$ it now suffices to show that $K \geq 4$. 

The size of the critical sphere is \(|S_{δ,t}| = \sum_{i=0}^{2} 2^i \binom{n}{i} = 2n^2 + 1\). We have

\[
K = \left\lfloor \frac{|S_{δ,t}||C|}{q^n} \right\rfloor \geq \left\lfloor \frac{(3^m + 1)^2}{2 \times 3^{m+1}} \right\rfloor,
\]

which is growing with respect to \(m\). As \(m = 3\) yields \(K = 4\), we see that \(C\) is optimal if \(m \geq 3\).

**Example 5.4** Let \(q = 2a\), where \(a \geq 3\) is an integer. Then a code \(C\) with \(n = 2^m\), \(|C| = a^n 2^{n-m-1}\) and \(d_{E_{\text{min}}}^2 = d_E^2(\gamma_2)\) exists, for any integer \(m \geq 2\). It can be constructed as \(C = C1 + 2C2\) where \(C1\) is a binary extended Hamming code and \(C2\) is \(\mathbb{Z}_2^n\). The bound described in this work will be used to show that \(C\) has the highest \(d_{E_{\text{min}}}^2\) possible.

Consider the inner distance measure \(δ\) described in (4.5), with \(i = 1\), and \(t = \delta(\gamma_1)\). These \(δ\) and \(t\) are not necessarily the best possible, but they will be sufficient. The size of the critical sphere is \(|S_{δ,t}| = \sum_{i=0}^{1} 2^i \binom{n}{i} = 2n + 1\). We have

\[
K = \left\lfloor \frac{|S_{δ,t}||C|}{q^n} \right\rfloor \geq \left\lfloor \frac{2^{m+1} + 1}{2m+1} \right\rfloor = 2.
\]

The chosen \(δ\) results in \(\tilde{y} = (y_1, y_{q-1}, y_0, y_0, \ldots, y_0)^T\) and

\[
f_δ(\tilde{y}) = \frac{2Kd_E^2(\gamma_1) - d_{E}^2(\gamma_1)}{2\delta(\gamma_1)}.
\]

The bound becomes

\[
\frac{2t f_δ(\tilde{y})}{K - 1} = \frac{2Kd_E^2(\gamma_1) - d_{E}^2(\gamma_1)}{K - 1},
\]

which for \(K = 2\) simplifies to \(d_{E}^2(\gamma_2)\).

Hence, \(C\) is optimal with respect to \(d_{E_{\text{min}}}^2\).

**Example 5.5** Consider \(q = 4a\), where \(a \geq 2\) is an integer. Then a code \(C\) with \(n = 2^m\), \(|C| = a^n 2^{n-m-2}\) and \(d_{E_{\text{min}}}^2 = 4d_E^2(\gamma_1)\) exists, for any integer \(m \geq 2\). It can be constructed as \(C = C1 + 2C2 + 4C3\) where \(C1\) is a binary extended Hamming code, \(C2\) is a binary code with a parity check bit and \(C3\) is \(\mathbb{Z}_2^n\). The bound described in this work will be used to show that for large \(m\) the code \(C\) has the highest \(d_{E_{\text{min}}}^2\) possible.

Proceed as in Example 5.3, including the choice of \(δ\) and \(t\) to show that \(K \geq 4\) is sufficient.

The size of the critical sphere is \(|S_{δ,t}| = \sum_{i=0}^{2} 2^i \binom{n}{i} = 2n^2 + 1\). We have

\[
K = \left\lfloor \frac{|S_{δ,t}||C|}{q^n} \right\rfloor \geq \left\lfloor \frac{2^{2m+1} + 1}{2m+2} \right\rfloor,
\]

which is growing with respect to \(m\). As \(m = 3\) yields \(K = 4\), we see that \(C\) is optimal if \(m \geq 3\).

**Example 5.6** Consider \(q = 4a\), where \(a \geq 2\) is an integer. Then a code \(C\) with \(n = 2^m\), \(|C| = a^n 2^{2n-2m-1}\) and \(d_{E_{\text{min}}}^2 = 6d_E^2(\gamma_1)\) exists, for any even integer \(m \geq 4\). It can be constructed as \(C = C1 + 2C2 + 4C3\) where \(C1\) is a binary extended Preparata code, \(C2\) is a binary code with a parity check bit and \(C3\) is \(\mathbb{Z}_2^n\). The bound described in this work will be used to show that for large \(m\) the code \(C\) has the highest \(d_{E_{\text{min}}}^2\) possible.

Proceed as in Example 5.3, including the choice of \(δ\), but let \(t = 3δ(\gamma_1)\). With this value of \(t\), Theorem 4.5 gives that the bound must be one of the following values (given in order of size): \(2d_{E}^2(\gamma_1)\), \(2d_{E}^2(\gamma_2)\), \(2d_{E}^2(\gamma_3)\), \(2d_{E}^2(\gamma_4)\).
\[d_E^2(\gamma_2), 4d_E^2(\gamma_1), 2d_E^2(\gamma_1) + d_E^2(\gamma_2), 6d_E^2(\gamma_1), 2d_E^2(\gamma_2), 4d_E^2(\gamma_1) + d_E^2(\gamma_2), 2d_E^2(\gamma_1) + 2d_E^2(\gamma_2), 3d_E^2(\gamma_2).\]

Note that the order of size here depends on \(q \geq 5\). Using that \(q \geq 8\), so that \(d_E^2(\gamma_2) \geq (2 + \sqrt{2})d_E^2(\gamma_1)\), it can be seen that

\[f_\delta(\gamma) \leq \frac{(2 - \sqrt{2})(K - 2) + 1}{2\delta(\gamma)}d_E^2(\gamma_2),\]

so that the bound \((2tf_\delta(\gamma)/(K - 1))\) is less than or equal to

\[\frac{3(2 - \sqrt{2})(K - 2) + 1}{K - 1}d_E^2(\gamma_2),\]

which is less than \(2d_E^2(\gamma_2)\) when \(K \geq 7\). Hence, whenever \(K \geq 7\), the bound is \(6d_E^2(\gamma_1)\).

The size of the critical sphere is \(|S_{\delta, t}| = \sum_{i=0}^3 2^i \binom{n}{i} \geq n^3\), with equality if and only if \(n = 3\). We have

\[K = \left[ \frac{|S_{\delta, t}|}{C} \right] \approx \left[ \frac{2^{3m}}{2^{2m+1}} \right],\]

which is growing with respect to \(m\). As \(m = 4\) yields \(K = 8\), we see that \(C\) is optimal if \(m \geq 4\).

**Example 5.7** Consider \(q = ap\), where \(p \geq 5\) is a prime and \(a\) is a positive integer. Then a code \(C\) with \(n = p^m - 1, |C| = a^n p^{n-(r-1)m-1}\) and \(d_{E_{\min}}^2 = 2rd_E^2(\gamma_1)\) exists, for \(m \geq 1\) and \(r \leq (p - 1)/2\), both integers. It can be constructed as \(C = C1 + pC2\) where \(C1\) is a code from Roth & Siegel (1994) and \(C2 = Z_q^n\). Remark that the code \(C1\) was originally developed to perform well with respect to the Lee metric. The bound described in this work will be used to show that for large enough \(m\) the code \(C\) has the highest \(d_{E_{\min}}^2\) possible.

In the case \(r = 2\), assume \(m \geq 2\) and proceed as in Example 5.3, including the choice of \(\delta\) and \(t\) to show that \(K \geq 4\) is sufficient. The size of the critical sphere is again \(2n^2 + 1\), and

\[K = \left[ \frac{2(p^m - 1)^2 + 1}{p^{m+1}} \right] \geq \left[ \frac{p^{2m}}{p^{m+1}} \right] \geq p,\]

which is sufficient to claim \(K \geq 4\), so that \(C\) is optimal with respect to \(d_{E_{\min}}^2\).

In the case \(r = 3\), assume \(m \geq 2\) and proceed as in Example 5.6, including the choice of \(\delta\) and \(t\) to show that \(K \geq 7\) is sufficient. Note that \(r = 3\) requires \(p \geq 7\). The size of the critical sphere is again at least \(n^3\), and

\[K = \left[ \frac{(p^m - 1)^3}{p^{2m+1}} \right] = \left[ \frac{p^{3m}(p^m - 1)^3}{p^{3m}p^{2m+1}} \right] \geq \left( \frac{p^{m-1}}{p^m} \right)^3 p.\]

But \(p^m \geq 49\), so \(((p^m - 1)/p^{m})^3 \geq 0.94\), which multiplied by \(p\) is at least 6.58, so \(K \geq 7\) is obtained. Hence, \(C\) is optimal with respect to \(d_{E_{\min}}^2\).

In the general case, let \(\delta\) be defined as in (4.5) with \(i = 1\) and let \(t = rd_E^2(\gamma_1)\).

To bound from above the size of the critical sphere some preparation is necessary. First, note that \(n = p^m - 1 \geq (2r + 1)^m - 1 > 4r^2\). Next, we will need

\[
\binom{n}{r} \geq \frac{n^n - \binom{n}{r} r^{r-1}}{r!}.
\]

That this holds can be seen in the following manner: write \(\binom{n}{r}\) as a polynomial in \(n\). The denominator is \(r!\), while the numerator is \(n(n-1)\cdots(n-r+1)\). When expanding this polynomial, we note that the
signs are alternating. Also, we will show that \( r^2|c_k| > |c_{k-1}| \), where \( c_k \) is the coefficient of \( n^k \) for any \( k \).\(^1\) We see this by the following equation

\[
r^2|c_k| = r^2 \sum_{i} p_k^{(i)} = r \sum_{i} \sum_{j=0}^{r-1} p_k^{(i)} > \sum_{i} \sum_{j=0}^{r-1} j p_k^{(i)} > \sum_{i} p_k^{(i)} = |c_{k-1}|,
\]

where \( p_k^{(i)} \) for different \( i \) span over all products of \( r - k \) distinct integers from the interval \([0, r-1]\). As \( n > r^2 \), we get \( |c_k| n^k > |c_{k-1}| n^{k-1} \). Now (5.1) follows easily as

\[
\binom{n}{r} = \frac{\sum_k c_k n^k}{r!} \geq \frac{c \gamma^r + c_{r-1} n^{r-1}}{r!} = \frac{n^r - \binom{r}{2} n^{r-1}}{r!},
\]

where the inequality follows from pairing adjacent terms.

Using (5.1), the size of the critical sphere can be estimated as

\[
|S_{\delta,r}| = \sum_{i=0}^{r} 2^i \binom{n}{i} \geq 2^r \binom{n}{r} \geq \frac{2^r}{r!} \left( n^r - \binom{r}{2} n^{r-1} \right).
\]

Using that \( m \geq 2 \) and that only cases where \( r \geq 4 \) so that \( p \geq 11 \) need to be dealt with,

\[
|S_{\delta,r}| \geq \frac{1.98^r}{r!} \left( p^{rm} - \binom{r}{2} p^{(r-1)m} \right) \geq \frac{1.98^r}{r!} p^{rm-2} \left( p^2 - \binom{r}{2} \right)
\]

\[
\geq \frac{1.98^r}{r!} p^{rm-2} \left( (2r+1)^2 - \binom{r}{2} \right) > \frac{1.98^r}{(r-1)!} p^{rm-2} (3r+4).
\]

We get

\[
K \geq \left[ \frac{1.98^r (3r+4) p^{m-2}}{(r-1)! p^{(r-1)m+1}} \right] = \left[ \frac{1.98^r (3r+4) p^{m-3}}{(r-1)!} \right],
\]

which is strictly increasing with respect to \( m \), and \( K \to \infty \) when \( m \to \infty \) and \( r \) is fixed. The bound has the form

\[
\frac{2t \delta \gamma}{K-1} = \frac{2(K-1) d_E^2(\gamma_1) + (d_E^2(\gamma_2) - 2d_E^2(\gamma_2)) r}{K-1}.
\]

The bound quite clearly approaches \( 2rd_E^2(\gamma_1) \) from above as \( K \) approaches infinity, so for any fixed \( r \), the bound will be tight for large enough \( m \).

The code \( C \) is optimal with respect to \( d_E^2 \) if \( m \) is large enough. For \( r = 2 \) and \( r = 3 \) we have seen that \( m \geq 2 \) is sufficient.

6. Conclusion

We have considered Elias’ sphere argument for forming upper bounds on codes over GU signal sets (Berlekamp (1968) pp. 318–321 and MacWilliams & Sloane (1977) pp. 558–564), taken an improvement of it developed for symmetric PSK with squared Euclidean distance (Nilsson et al. (2008); Laksman et al. (2010),

\(^1\)In fact, \(|c_k|\) are Sterling numbers of the first kind, the number of ways in which \( r \) distinct objects can be placed in \( r - k \) cycles. The property shown for them here may be – and probably is – known.
2009a,b)) and generalized the improvement to work also for asymmetric PSK and other additive and translation-invariant distance measures.

Codes over asymmetric PSK may be interesting for applications since there are examples of code parameters where codes over asymmetric PSK have higher minimum Euclidean distance than codes over symmetric PSK, although the difference seem not to be large. The bound, as well as the proof of it, is a generalization of a previous result Laksman et al. (2010) for codes over symmetric PSK. The bound is formulated in terms of extremal columns, but as the formulation is more general, less can be said about the extremal columns. Nonetheless, results to help determining the extremal columns for given parameters are presented. The results in the paper are general, valid for both symmetric PSK and asymmetric PSK block codes, and most of the results can be applied for any additive and translation-invariant distance measure.

Noise in wireless communication is often assumed to be AWGN, which leads to the codeword which has the least square Euclidean distance to the received signal being the codeword most likely sent. Typically, the error-correcting codes used are created in order to give the best error-correcting capability, i.e., the greatest minimum squared Euclidean distance, in relation to the amount of data that has to be transmitted. If the noise follows a different distribution than AWGN, ML decoding is no longer equivalent to finding the codeword or codesequence closest with respect to squared Euclidean distance to the received signal. A different distribution of noise gives rise to a different distance measure to use when measuring proximity between signals.

A method for forming upper bounds on minimum squared Euclidean distance for PSK block codes has been generalized to work for other distances as well, and special consideration has been given to the distances described by (2.1), a family which contains squared Euclidean distance. Several classes of codes have by use of the bound been shown to be optimal with respect to $d_{E_{\text{min}}}^2$ within the set of block codes over symmetric PSK.

It is of general importance to understand the limits of minimum distance (with respect to a relevant distance measure) for a code with certain parameters in order to maximize the potential for error correction and noise independence. This approach, based on an extended version of the Elias’ bound, has the advantage of being independent of the structure of the code – a code is only assumed to be a subset of the universe.

We consider the ease with which generalizations and extensions can be made, demonstrated by the results of the paper, as an important virtue of the present approach.

REFERENCES


IEEE Journal on Selected Areas in Communications, 7 (6), 900 – 913.


Optimal Computer Crash Performance Precaution

Efraim Laksman, Hakan Lennerstad, Lars Lundberg
Optimal Computer Crash Performance Precaution

Efraim Laksman†  Håkan Lennerstad‡  Lars Lundberg§

Blekinge Institute of Technology, Sweden

received 18th April 2011, revised 19th December 2011, accepted 18th January 2012.

For a parallel computer system with \( m \) identical computers, we study optimal performance precaution for one possible computer crash. We want to calculate the cost of crash precaution in the case of no crash.

We thus define a tolerance level \( r \) meaning that we only tolerate that the completion time of a parallel program after a crash is at most a factor \( r + 1 \) larger than if we use optimal allocation on \( m - 1 \) computers. This is an \( r \)-dependent restriction of the set of allocations of a program. Then, what is the worst-case ratio of the optimal \( r \)-dependent completion time in the case of no crash and the unrestricted optimal completion time of the same parallel program?

We denote the maximal ratio of completion times \( f(r, m) \) - i.e., the ratio for worst-case programs. In the paper we establish upper and lower bounds of the worst-case cost function \( f(r, m) \) and characterize worst-case programs.

Keywords: parallel computer, scheduling, computer crash, load balancing, process allocation, optimization

1 Introduction

Availability and performance are important in computer systems, and clusters of computers are often used to obtain both Pfister (1998). In order to obtain high performance we want to distribute the work load as evenly as possible between the computers in the cluster. High availability is obtained by redistributing the workload when computers break down. Most high availability systems are designed to handle one failure, i.e., the system should redistribute the load from a crashed computer to the other computers. Multiple failures are extremely unlikely and the designers of high availability systems are in most cases satisfied if their system can continue to operate in case one computer goes down.

When a computer goes down, the processes on the crashed computer must be redistributed to the other computers. One would like to avoid excessive redistribution of processes. Therefore, we would like to move only the processes on the crashed computer, i.e., the processes on the other computers should not be redistributed.

In Graham (1969) a tight upper bound is established for \( w_L/w_0 \), where \( w_L \) is optimal allocation with a restricted allocation set, and \( w_0 \) is the similar quantity with unrestricted allocation. The allocation restriction is in terms of a certain allocation algorithm. A similar but improved result, due to an improved

---

†Email: efl@bth.se
‡Email: hln@bth.se
§Email: llu@bth.se

1365–8050 © 2012 Discrete Mathematics and Theoretical Computer Science (DMTCS), Nancy, France
algorithm, is given in Coffman Jr. et al. (1978). Finally, a polynomial time approximation scheme to multiprocessor scheduling has been presented by Hochbaum and Shmoys (1987). In the present paper, however, the restriction is not given by an algorithm, but by the specified crash deterioration tolerance. The results are independent of specific algorithms, so they are valid for all algorithms.

There has been a lot of research on performance of multiprocessors involving processor failures. In Girault et al. (2009) multiprocessor performance is optimized by a new method according to completing time and reliability of schedule when computer failures occur under certain restrictions. Ever et al. (2009) study the performance of Beowulf clusters under the assumptions of exponential failure and repair rates. The paper Al-Rousan and Shaout (2004) investigates the performance of large scale multiprocessors assuming a Weibull failure process. In contrast, the present paper assumes no statistical distribution for failures, nor any specific multiprocessor architecture. Furthermore, the performance issue is turned around in the sense that the accepted tolerance to performance loss in case of a failure is the starting point, and the result is the performance in case of no failure. This formulation of the problem appears so far to be rather unique in the field.

Note that the approach taken here differs from reoptimization. In reoptimization, the solution to a problem is solved more efficiently, in some respect, by exploiting the solution of a similar problem. This is not the case in the present research, since here the result consists in comparing two optimal solutions - how much deterioration that may occur in the case of no crash if certain well specified precautions are taken if expecting a computer crash. The computation of one of them is not taken advantage of to calculate the other. The present results do not concern how to calculate the bounds, in fact they involve NP-hard problems (see Section 5).

Many (real-time) systems have strict performance requirements that must be met also when one computer is down. The critical aspect in such systems is to keep the load on the most heavily loaded computer under a certain threshold. Since we only redistribute the processes on the crashed computer, the performance when one computer is down may depend on the identity of the crashed computer. For instance, consider a simple case with three identical computers and four processes: process one and two have a performance requirement of two performance units (e.g. MIPS) each, and processes three and four have a performance requirement of one unit each. Process one executes on computer one, and process two executes on computer two, whereas processes three and four share computer three. This means that under normal conditions when all computers are up and running the load is evenly balanced, e.g. the load is two on each computer. We will refer to the case when all computers are up and running as the normal-case scenario. If computer one or two goes down the load on some computer will be four, but if computer three goes down it is possible to put process three on computer one and process four on computer two and in that case the load on the most heavily loaded computer is only three. We will refer to the case when the most unfavorable computer is down as the worst-case scenario.

The optimal normal-case and optimal worst-case allocations are often not compatible, i.e., one can in most cases not obtain optimal worst-case performance and optimal normal performance based on the same allocation of processes to computers. In the case above with three computers and four processes, one can obtain optimal worst-case performance by allocating processes one and three to computer one and processes two and four to computer two and leave computer three empty. If computer one or two goes down, the load on the crashed computer is simply moved to computer three. However, using that allocation the normal-case performance will be the same as the worst-case performance. If we spread out the load as evenly as possible we obviously get better normal-case performance, but as we discussed above the worst-case performance is lower since the load on the most heavily loaded computer could be
as high as four.

After a computer break down we assume that all processes on that computer are reallocated optimally on the other computers. We also assume the worst possible break down, i.e., by the computer that maximizes the resulting completion time after reallocation.

In this paper a tolerance level parameter $r$ is introduced. We tolerate that the worst-case completion time after a computer crash is within a factor $1 + r$ from the optimal completion time on $m - 1$ computers – i.e., when we allocate optimally assuming that one computer break down. Only allocations that fulfill this condition are considered. This is $r$-restricted allocation – we optimize the completion time of a program in the normal case within this restricted set of allocations. Note that $0 \leq r \leq 1$ where $r = 0$ means that we allow no deterioration after a crash, no tolerance, and $r = 1$ represents no restriction on the set of allocations.

We compare two normal-case completion times: $r$-restricted completion time to non-restricted. For a parallel program $P$, the function $f(P, r, m) \geq 1$ denotes the ratio of these two quantities. If $A$ denotes an allocation, $A_P$ the program $P$ allocated by $A$, and $T(A_P)$ the completion time of $A_P$, we thus define

\[ f(P, r, m) = \frac{\min_A r\text{-restricted } T(A_P)}{\min_A T(A_P)}. \]

Furthermore, we consider worst-case programs, i.e., programs $P$ maximizing $f(P, r, m)$:

\[ \hat{f}(r, m) = \max_P f(P, r, m). \]

In this paper we present upper and lower bounds on $\hat{f}(r, m)$. The upper bound of $\hat{f}(r, m)$ is deduced by identifying a certain subset of worst-case programs, and taking advantage of the properties of these programs. The lower bound is deduced by an entirely different method. It is derived from the properties of certain specific programs. The bounds meet for some values of $m$ and $r$.

A related problem has previously been studied in Lundberg and Svahnberg (2003). In that case, the authors defined an algorithm for allocating processes to computers so that the performance is at most a factor 1.5 of the optimal performance for the normal-case scenario while at the same time guaranteeing that the performance when one computer has crashed is withing a factor of 1.5 of the optimal performance of the worst-case scenario. This means that from the existence of this algorithm we know that $\hat{f}(0.5, m) \leq 1.5$.

The problem of optimizing the performance of the worst-case scenario when more than one computer crash has been studied by Klonowska et al. for the special case when there is only one process on each computer, and this process requires one performance unit Klonowska et al. (2005). In this scenario the reallocations of processes on a crashed computer is done on one specific running computer according to a global recovery scheme. This scheme provides maximum load balancing if it is designed according to a so called Golomb ruler. It turns out that the scheme can be further improved by means of a modular Golomb ruler Klonowska et al. (2005).

Reliability in multiprocessor systems has been studied previously, e.g. in Chen and Cherkassky (1992). In that paper a static schedule based on rotations of an optimal schedule for the normal case is used, and the probability that all processes are run at least once is given as a function of the number of processors, number of rotations used and probability of break down for a single processor. Note the differences between Chen and Cherkassky (1992) and the present paper: here we assume the existence of at most one
computer breakdown, allow reallocation of processes from a computer which has broken down, consider optimal schedules, which may be hard to find, rather than a specific schedule, and examine completion time, rather than the probability of complete break down.

2 Mathematical formulation of the problem

By the term *normal case* we mean no crash and no restriction on the set of allocations for a program. We start by defining notation.

- $m$ is the number of identical computers in the cluster.
- $P$ is a parallel program, *i.e.*, a set of independent processes.
- $A_P$ is an allocation of $P$, *i.e.*, some way of distributing the processes in $P$ among $m$ computers.
- $T_i(A_P)$ is the completion time of the $i$:th computer of program $P$ with allocation $A$.
- $T(A_P)$ is the time required to run $A_P$, *i.e.*, $T(A_P) = \max_{i=1,\ldots,m} T_i(A_P)$.
- $W(P)$ is the total work required to complete $P$, *i.e.*, $W(P) = \sum_{i=1}^m T_i(A_P)$.
- $\Phi_P$ denotes an allocation of $P$ which is optimal in the normal case, hence $T(\Phi_P) \leq T(A_P)$ for any allocation $A$.

The notation chosen is not standard, but it is easy to read, avoiding an overflow of subscripts and superscripts, even when considering workload on different computers for different allocations of different programs at the same time.

We next describe the assumptions on a computer crash. Having a program $P$ allocated with $A_P$, we assume that one of the $m$ computers crash. The first assumption is that the processes on the computer that goes down are distributed on the remaining computers in an optimal manner: to minimize global execution time. Furthermore, there are $m$ different computers that may crash. The second assumption is that the computer that crashes is the one that increases the completion time the most. A modification of $A_P$ from $m$ to $m-1$ computers according to these assumptions is denoted by $A'_P$.

We have thus defined the mapping $A_P \rightarrow A'_P$. In the allocation $A'_P$, two assumptions are built in: the crash is worst possible, and it is a best possible reallocation of $A_P$.

We thus have two optimality concepts: one with a crash and one without, which requires two more definitions:

- $A'_P$ is the best reallocation of $A_P$ given the worst-case scenario.
- $\Psi_P$ is an allocation of $P$ where $\Psi'_P$ is optimal: $T(\Psi'_P) \leq T(A'_P)$ for any allocation $A$.

There may be several allocations of $P$ that are optimal in the normal case. If it is so, then $\Phi_P$ should be chosen among these allocations as the one which results in $T(\Psi'_P)$ being minimal.

Similarly, there may be several allocations of $P$ that are optimal in the worst case. If it is so, then $\Psi_P$ should be chosen among these allocations as the one which results in $T(\Psi_P)$ being minimal.

Using these definitions, the $r$-restriction on an allocation $A_P$ is the following condition on $A'_P$,

$$T(A'_P) \leq T(\Psi'_P)(1 + r) .$$
Within this set of allocations, we are interested in one that gives minimal normal-case completion time \(T(A_P)\), i.e., an allocation \(A\) where

\[
\min_A \{ T(A_P) : T(A'_P) \leq T(\Psi'_P)(1 + r) \}.
\]

The function \(f\) is the ratio to the normal-case completion time, so we define

\[
f(P, r, m) = \min_A \{ T(A_P) : T(A'_P) \leq T(\Psi'_P)(1 + r) \} \quad \frac{T(\Phi_P)}{T(\Phi_P)}.
\]

We seek worst-case behaviour of \(f\) over all programs \(P\)

\[
\hat{f}(r, m) = \max_P f(P, r, m).
\]

It is immediate from the problem formulation that \(\hat{f}(r, m)\) is non-increasing in the parameter \(r\).

We next define terminology that will be used in this paper.

**Definition 1** A computer in the computer cluster is critical for the allocation \(A\) if it finishes its execution at \(T(A_P)\). An allocation is called a box allocation (or just “box”) if all computers are critical. A program \(P\) is called a worst program for \(r\) and \(m\) if \(f(P, r, m) = \hat{f}(r, m)\). An allocation \(A_P\) for a program \(P\) is called undominated if and only if there is no allocation \(B_P\) with

\[
\begin{align*}
T(B_P) & \leq T(A_P) \text{ and } T(B'_P) < T(A'_P), \text{ or} \\
T(B_P) & < T(A_P) \text{ and } T(B'_P) \leq T(A'_P).
\end{align*}
\]

An allocation which is not undominated is called dominated.

For all \(r\) and \(P\) there is an undominated allocation \(A_P\) that minimizes

\[
\{ T(A_P) : T(A'_P) \leq T(\Psi'_P)(1 + r) \}.
\]

To show this, assume that a dominated allocation \(B_P\) minimizes (3) for some \(r\). Since \(B_P\) is dominated, there exists an undominated allocation \(A_P\) with \(T(A_P) \leq T(B_P)\) and \(T(A'_P) \leq T(B'_P)\). Obviously also \(A_P\) minimizes (3).

If there are two undominated allocations \(A\) and \(B\) with \(T(A_P) = T(B_P)\) and \(T(A'_P) = T(B'_P)\), we disregard from one of them. It follows now from the definition of undominated allocations that we have a certain set of undominated allocations

\[
\{ \Phi_P, A_{P,1}, A_{P,2}, \ldots, A_{P,n-2}, \Psi_P \}
\]

with only strict inequalities as follows

\[
\begin{align*}
T(\Phi_P) & < T(A_{P,1}) < \ldots < T(A_{P,n-2}) < T(\Psi_P) \leq \\
& \leq T(\Psi'_P) < T(A'_{P,n-2}) < \ldots < T(A'_{P,1}) < T(\Phi'_P),
\end{align*}
\]

\(\text{(4)}\)
except possibly in \( T(\Psi_P) \leq T(\Psi'_P) \). This gives the following characterization of the function \( f \):

\[
f(P, r, m) = \begin{cases} 
\frac{T(\Phi_P)}{T(\Psi_P)} = 1, & 1 + r < \frac{T(\Phi_P)}{T(\Psi_P)} \\
\frac{T(\alpha P)}{T(\Psi_P)} = 1 + r, & \frac{T(\alpha P)}{T(\Psi_P)} \leq 1 + r < \frac{T(\Phi_P)}{T(\Psi_P)} \\
\frac{T(\alpha P_2)}{T(\Psi_P)} = 1 + r, & \frac{T(\alpha P_2)}{T(\Psi_P)} \leq 1 + r < \frac{T(\alpha P_1)}{T(\Psi_P)} \\
\ldots & \\
\frac{T(\Psi_P)}{T(\Psi_P)} = 1 + r, & 1 + r < \frac{T(\alpha P, P - 2)}{T(\Psi_P)} 
\end{cases}
\]

Note also that \( T(\Phi_P) \leq 2T(\Psi_P) \) is a trivial relationship, which in a way limits how much “unequal” the inequalities in (4) can be.

3 Upper bound on \( \hat{f} \)

We start by establishing some basic properties which are true for at least some worst program \( P \) for each pair of \( r \) and \( m \).

Lemma 1 For every \( r \) and \( m \), there is a worst program \( P \) such that \( \Phi_P \) or \( \Psi'_P \) is a box.

Proof: Assume that there is a program \( P \) such that neither \( \Phi_P \) nor \( \Psi'_P \) is a box. This means that there is at least one computer in each of the two allocations which is not critical. We choose a non-critical computer in \( \Phi_P \) and call it \( a_0 \) and a non-critical computer in \( \Psi'_P \) and call it \( a_1 \). We now create the program \( Q \) with the same processes as \( P \) and an extra process \( \alpha \) with \( T(\alpha) = \min \{ T(\Phi_P) - T(a_0), T(\Psi'_P) - T(a_1) \} \).

We have that \( T(\Phi_P) = T(\Phi_Q) \) and \( T(\Psi'_P) = T(\Psi'_Q) \). We also have that for any allocation \( A_Q \), one can form an allocation \( \alpha P \) with \( T(\alpha P) \leq T(A_Q) \) and \( T(\alpha P') \leq T(A'_Q) \), by allocating all processes from \( P \) to the same computer as they are allocated in \( A_Q \). It is now clear from the definition of \( f \) that \( f(P, r, m) \leq f(Q, r, m) \), so that if \( P \) is a worst program, then so is \( Q \).

If there are several non-critical computers to start with, we may have to repeat the argument several times. Eventually we get a worst program for which \( \Phi_P \) or \( \Psi'_P \) is a box. The lemma is proved. \( \square \)

Lemma 2 If \( P \) is a worst program for \( r < \frac{m-2}{m} \) and \( \Psi'_P \) is a box, then also \( \Phi_P \) is a box.

Proof: Assume that \( \Psi'_P \) is a box and that \( \Phi_P \) is not a box. We prove that then \( P \) cannot be a worst program since a specific program \( Q \) has higher value of \( f \).

By (4), we always have \( f(P, r, m) \leq \frac{T(\Phi_P)}{T(\Psi_P)} \). We will have \( T(\Phi_P) \leq T(\Psi'_P) = \frac{W(P)}{m-1} \), as \( \Psi'_P \) is a box. We will also have \( T(\Phi_P) > \frac{W(P)}{m} \), as \( \Phi_P \) is not a box.

Now we have that \( f(P, r, m) \leq \frac{T(\Psi_P)}{T(\Phi_P)} \leq \frac{W(P)}{W(P) / (m-1)} = \frac{m}{m-1} \).

We next study the program \( Q \) with \( m-1 \) processes of length 1 and \( m-1 \) processes of length \( m-1 \). Then \( T(\Phi_Q) = m - 1 \) and \( T(\Psi'_Q) = 2(m-1) \), and \( T(\Psi'_Q) = T(\Psi_Q) = m \). Since \( T(\alpha P) \geq m \) for any other allocation \( A \), it is clear that \( \Phi_Q \) and \( \Psi_Q \) are the only undominated allocations of \( Q \). The condition \( T(\alpha P) \leq T(\Psi_P) + 1 + r \) with \( A = \Phi \) now gives the \( r \)-values where \( \Phi \) is allowed. We get

\[
f(Q, r, m) = \begin{cases} 
\frac{m - 1}{m - 1}, & r < \frac{m-2}{m} \\
1, & r \geq \frac{m-2}{m} 
\end{cases}
\]
Thus $P$ cannot be a worst program for $r < \frac{m-2}{m}$, since for these values $f(P, r, m) < f(Q, r, m)$.

**Corollary 3** If $r < \frac{m-2}{m}$, there is a worst program $P$ such that $\Phi_P$ is a box.

**Proof:** This follows directly from Lemma 1 and 2.

**Lemma 4** If $P$ is a worst program for $r \geq \frac{m-2}{m}$ and $\Phi_P$ is a box, then $f(P, r, m) = 1$.

**Proof:** Assume that $T(\Phi_P) = 1$. We know that $\Phi_P$ is a box allocated on $m$ computers, so $W(P) = m$. We know also that $\Psi'_P$ is allocated on $m - 1$ computers only, so by $W(P) = m$ we have $T(\Psi'_P) \geq \frac{m}{m-1}$.

For any program $P$, if $\frac{T(\Phi'_P)}{T(\Psi'_P)} \leq 1 + r$ the allocation $\Phi_P$ is allowed so $f(P, r, m) = 1$. We get the implication $f(P, r, m) > 1 \Rightarrow 1 + r < \frac{T(\Phi'_P)}{T(\Psi'_P)}$. Using $T(\Phi'_P) \leq 2T(\Phi_P) = 2$ and $T(\Psi'_P) \geq \frac{m}{m-1}$ now gives us

$$f(P, r, m) > 1 \Rightarrow 1 + r < \frac{T(\Phi'_P)}{T(\Psi'_P)} \leq \frac{2(m-1)}{m}.$$  

We can also express the implication as $f(P, r, m) > 1 \Rightarrow r < \frac{m-2}{m}$, which is equivalent to the statement of the lemma.

**Corollary 5** If $r \geq \frac{m-2}{m}$ and $f(P, r, m) > 1$, there is a worst program $P$ such that $\Psi'_P$ is a box.

**Proof:** This follows directly from Lemma 1 and 4, and from that $f(P, r, m) \geq 1$ for all $P$, $r$ and $m$ where it is defined.

We next present the upper bounds of $\hat{f}$ in two theorems.

**Theorem 6** For any $m$, we have that

$$\hat{f}(r, m) < \frac{2}{1 + r}, \text{ for } 0 \leq r < 1.$$

**Proof:** For any program $P$, we have

$$\hat{f}(r, m) \leq \frac{T(\Psi_P)}{T(\Phi_P)} \leq \frac{T(\Psi'_P)}{T(\Phi'_P)} \leq \frac{1 + r}{T(\Phi'_P)} \geq \frac{2}{T(\Phi_P)} \geq \frac{1 + r}{T(\Phi'_P)}.$$  

The inequalities (5) and (7) are trivial. The inequality (6) follows from that we would have $f(P, r, m) = 1$ otherwise.

Joining these inequalities we get

$$\hat{f}(r, m) \leq \frac{T(\Psi'_P)}{T(\Phi_P)} \leq \frac{2}{T(\Phi'_P)} < \frac{2}{1 + r}.$$  

51
From the proof of this theorem, we see that if a worst program is to meet the given bound, it must contain at least one process of length \( T(\Phi_P) \) and \( \Phi_P \) must be a box. But for \( r \geq \frac{m-2}{m} \) we know that \( \Phi_P \) is not a box, so we can get a stronger bound in this case.

**Theorem 7** For \( \frac{m-2}{m} \leq r \leq \frac{m-2}{m-1} \), we have \( \hat{f}(r, m) \leq \frac{m-2}{r(m-1)} \), and for \( \frac{m-2}{m-1} < r \), we have \( \hat{f}(r, m) = 1 \).

**Proof:** Since \( r \geq \frac{m-2}{m} \), we have that for a worst program \( P \), \( \Phi_P \) is not a box. This means that in \( \Phi_P \), there will be a computer which runs for the minimum amount of time, \( x < T(\Phi_P) \). We immediately get \( T(\Phi_P) + x \geq T(\Phi_P)' \geq (1 + r)T(\Psi_P)' \), as we would otherwise have \( \hat{f}(r, m) = 1 \), which is a trivial lower bound.

Since \( \Phi_P \) has one computer with run time \( T(\Phi_P) \), one computer with run time \( x \) and the other computers have run time of at least \( x \) and at most \( 1 \), we get \( (m-1)T(\Phi_P) + x \geq W(P) \geq T(\Phi_P) + (m-1)x \).

We will only need the lower bound on \( W(P) \). On the other hand, we have that \( \Psi_P' \) is a box. Distributing \( P \) equally on all \((m-1)\) computers, we get

\[
T(\Psi_P') = \frac{W(P)}{m-1} \geq \frac{T(\Phi_P)}{m-1} + x.
\]

We get

\[
T(\Phi_P) + x \geq (1 + r)T(\Psi_P') \geq (1 + r) \left( \frac{T(\Phi_P)}{m-1} + x \right).
\]

Solving for \( x \) now gives

\[
\frac{(m-2 - r)T(\Phi_P)}{r(m-1)} \geq x.
\]

But we have

\[
f(P, r, m) \leq \frac{T(\Psi_P')}{T(\Phi_P)} \leq \frac{1}{T(\Phi_P)} \frac{T(\Phi_P) + x}{1 + r}
\]

Replacing with our bound on \( x \), we get rid of \( T(\Phi_P) \), and get

\[
f(P, r, m) \leq \frac{1 + \frac{m-2-r}{r(m-1)}}{1 + r} = \frac{rm + m - 2 - 2r}{r(1 + r)(m-1)} = \frac{m - 2}{r(m-1)}.
\]

However, we know that \( \hat{f} \) is trivially bounded by 1 from below, so that the above reasoning cannot hold for \( r > \frac{m-1}{m-2} \), where \( \Phi_P \) must be permitted. \( \square \)
4 Lower bound on \(\hat{f}\)

The worst programs we have found so far provide our lower bounds. We remark that not all of our examples here result in either \(\Phi\) or \(\Psi\) being a box. The programs can be altered, according to the proof of Lemma 1, in such a way that \(\Phi\) or \(\Psi\) becomes a box, but this will only complicate calculation of the value of \(f\) (by introducing additional undominated allocations), without changing the value of \(\hat{f}\) in the region (in terms of \(r\) and \(m\)) where the given program gives an interestingly high value of \(\hat{f}\).

While Lemma 8 and Corollary 9 have not been explicitly used to find any bound on \(\hat{f}\), they have been helpful when searching for worst programs, and may be helpful to anyone intending to continue this line of research.

**Lemma 8** For any \(r\) and \(m\), there is a worst program \(P\) which does not have any pair of processes on the same computer in both allocations \(\Phi_P\) and \(\Psi'_P\).

**Proof:** Assume that there is a program \(P\) which has a pair of processes on the same computer in both allocations \(\Phi_P\) and \(\Psi'_P\), such that \(f(P, r, m) = \hat{f}(r, m)\). Then, consider the program \(Q\) which is identical to \(P\), except that the two processes allocated to the same computer in both \(\Phi_P\) and \(\Psi'_P\) are merged into one process of length equal to the sum of the two previous length.

We have that \(T(\Phi_P) = T(\Phi_Q)\) and \(T(\Psi'_P) = T(\Psi'_Q)\). We also have that for any allocation \(A_Q\), one can form an allocation \(A_P\) with \(T(A_P) = T(A_Q)\) and \(T(A'_P) \leq T(A'_Q)\) (by allocating processes in \(A_P\) just as they are allocated in \(A_Q\)). It is now clear from the definition of \(\hat{f}\) that \(f(P, r, m) \leq f(Q, r, m)\), so that if \(P\) is a worst program, then so is \(Q\).

If \(P\) carried large groups processes on the same computer in both \(\Phi_P\) and \(\Psi'_P\), then the argument above may have to be repeated several times, but eventually we will have a worst program with the sought after condition. \(\square\)

**Corollary 9** For every \(r\) and every \(m\), there is a worst program \(P\), such that every computer in \(\Phi_P\) executes at most \(m - 1\) processes and every computer in \(\Psi'_P\) executes at most \(m\) processes.

**Proof:** By the pigeon hole principle, if a computer in \(\Phi_P\) executes more than \(m - 1\) processes, then at least 2 of these processes must wind up on the same computer in \(\Psi'_P\), in which case the program is uninteresting according to the lemma above.

Similarly, if a computer in \(\Psi'_P\) executes more than \(m\) processes, then at least 2 of these processes must come from the same computer in \(\Phi_P\), in which case the program is uninteresting according to the lemma above. \(\square\)

### 4.1 First program

Consider the program \(P\) with \(k \geq 2\) processes of length 1 and \(m - 1\) processes of length \(k\), when we have \(m \geq k + 1\) computers. It is fairly easy to see that \(T(\Phi_P) = k, T(\Phi'_P) = 2k\) and \(T(\Psi_P) = T(\Psi'_P) = k + 1\). See Figure 1.

Clearly no other allocations can be undominated, as they would require at least as much time as \(\Psi_P\), even before a crash. Thus we get

\[
f(P, r, m) = \frac{k + 1}{k}, \quad \text{when } 1 + r < \frac{2k}{k + 1}.
\]
4.2 Second program

Consider the program $P$ with $k - 1$ processes of length 1 (we want $k \geq 2$), one process of length $k - 1 + \epsilon$, with $0 \leq \epsilon < 1$ and $m - 2$ processes of length $k - 1$, when we have $m \geq k + 1$ computers. It is fairly easy to see that $T(\Phi_P) = k - 1 + \epsilon$, $T(\Phi'_P) = 2(k - 1) + \epsilon$ and $T(\Psi_P) = T(\Psi'_P) = k$. See Figure 2.

Clearly no other allocations can be undominated, as they would require at least as much time as $\Psi_P$, even before a crash. Thus we get

$$f(P,r,m) = \frac{k}{k - 1 + \epsilon}, \text{ when } 1 + r < \frac{2(k - 1) + \epsilon}{k}.$$ 

“Solving” the condition for $\epsilon$, one attains $\epsilon > kr - k + 2$. We now set $\epsilon = kr - k + 2 + \epsilon'$, with $\epsilon' > 0$. This gives us

$$f(P,r,m) = \frac{k}{kr + 1 + \epsilon'}.$$ 

Now we let $\epsilon'$ approach 0, so that $f(P,r,m)$ gets arbitrarily close to $\frac{k}{kr+1}$.

4.3 Combining the programs

Since we have $\hat{f}(r,m) \geq f(P,r,m)$ for all programs $P$, we have

$$\hat{f}(r) \geq \max \left( \frac{k + 1}{k}, \frac{k}{kr + 1} - \epsilon \right), \text{ for } k \text{ such that } \frac{k - 2}{k} \leq r < \frac{k - 1}{k + 1},$$
for arbitrarily small $\epsilon > 0$. If we calculate which term the maximum function will “choose” for which $r$, we may express it as

$$\hat{f}(r, m) \geq \left\{ \begin{array}{ll}
\frac{k}{kr+1} - \frac{\epsilon}{k}, & k \leq r < k+1 \\
\frac{k-2}{k+1} - \frac{1}{k^{2+k}} & 1 \leq r \leq \frac{k-1}{k+1}
\end{array} \right.$$

given that $m$ is large enough, which it may be considered to be when

$$r < \frac{m - 2}{m} \Leftrightarrow m > \frac{2}{1 - r},$$

because this allows usage of the programs and allocations described above. In the case where $m \leq \frac{2}{1 - r}$, we may still use the second kind of program described, with $k = m$. This gives the following lower bound. For arbitrarily small $\epsilon$ we have

$$\hat{f}(r, m) \geq \frac{m}{mr + 1} - \epsilon, \text{ if } \frac{m - 2}{m} \leq r < \frac{m - 1}{m}.$$

We gather these bounds in a theorem:

**Theorem 10** We have

$$\hat{f}(r, m) \geq \left\{ \begin{array}{ll}
\frac{k}{kr+1} - \frac{\epsilon}{k}, & k \leq r < k+1 \\
\frac{k-2}{k+1} - \frac{1}{k^{2+k}} & 1 \leq r \leq \frac{k-1}{k+1}
\end{array} \right.$$

for $r \in \left[ 0, \frac{m - 2}{m} \right)$. 

and

$$\hat{f}(r, m) \geq \frac{m}{mr + 1} - \epsilon, \text{ for } r \in \left[ \frac{m - 2}{m}, \frac{m - 1}{m} \right),$$

and

$$\hat{f}(r, m) \geq 1, \text{ for } r \in \left[ \frac{m - 1}{m}, 1 \right),$$

for any $\epsilon > 0$.

## 5 Conclusions

When allocating processes to computers in a cluster there is a trade-off between normal-case and worst-case performance. In this paper we have been able to quantify this trade-off. The function $\hat{f}(r, m)$ makes it possible to determine how close to the optimal normal-case performance we can get, given that we accept a worst-case performance within a factor of $1 + r$ of optimal worst-case performance.

For instance, consider a cluster of at least 4 computers (i.e., $m \geq 4$) and a scenario where we accept worst-case performance of 1.5 times the optimal worst-case performance (i.e., $r = 0.5$). In that situation, Figure 3 shows that for every program it is possible to meet this restriction and obtain a normal-case performance of at most 1.33 . . . times the optimal normal-case performance. However, finding the optimal allocation $A$ for given parameters $P, r, m$ is NP-hard. This can be seen as in the case $r = 1$, finding $A$ is equivalent to finding $\Phi$ which contains the NP-complete problem of multiprocessor scheduling (on $m$ computers), while in the case $r < 1$, one must determine $T(\Psi_P)$, which also contains the problem of multiprocessor scheduling (on $m - 1$ computers). For proof of the NP-completeness, see e.g. Garey and Johnsson (1979).
Also reallocating optimally, i.e., finding the best $A'_P$ for a given $A_P$, is NP-hard. This can be seen by considering the case with 3 computers. No matter which computer crashes, reallocating the processes originally allocated to it corresponds to finding a 2-partition of the union of those processes and a process of length equal to the difference between the original load on the two other computers, a problem which is known to be NP-hard Garey and Johnsson (1979).

As can be seen in Figures 3 and 5 there is very little space between the upper and lower bound. The slack of the bounds is greatest in the proximity of $r = \frac{1}{6}$ for all $m$ and $r = \frac{1}{3}$ at $m = 3$. An open question is how the bounds can be improved.

6 Acknowledgements

The authors would like to thank reviewer B for comments regarding computational complexity and intractability and thank Prof. Bengt Aspvall for discussions on the same topic.
Fig. 4: Lower and upper bound of $\hat{f}$.

Fig. 5: The difference between the upper and lower bound of $\hat{f}$. 
References


Paper III

Map Matching and Hidden Markov Models

Efraim Laksman
Abstract
The present paper considers the problem of Map Matching (MM), i.e. matching time and location measurements to a route in a road network, and presents a probabilistic algorithm to solve it. The algorithm differs from previous probabilistic algorithms for MM mainly in that it is based on a second order Hidden Markov Model (HMM), as opposed to first order HMM. This allows a more detailed analysis of the data while preserving algorithmic complexity $O(n)$.

Both measurement densities and transition probabilities are determined with respect to Kolmogorov's third axiom, which in this context implies that the probabilities are additive over a partition of a road segment.

Keywords: Map Matching, Hidden Markov Model

1. Introduction

The problem of Map Matching (MM) is two-folded and may be described as follows: Let a vehicle travel on a road network and make measurements of the positions of the vehicle, where noise is included. MM can be viewed as the problem to determine the route taken by the vehicle. This view is relevant for example for purpose of using GPS for electronic toll collection. MM can also be viewed as the problem of determining the current position of the vehicle every time a new measurement of the current position is made. This view is relevant for real-time navigation purposes.

We present a probabilistic algorithm for MM. The structure of this algorithm is Viterbi's algorithm Viterbi (1967), hereafter referred to only as Viterbi. Viterbi produces the most likely route for a sequence of observations. Viterbi can also output the most likely current position every time
a new observation is made, and in particular it manages to do so in time which is constant with respect to the number of previous measurements. For the purpose of real-time navigation, this is the maximum complexity an algorithm may have.

Viterbi is commonly used with many previous probabilistic algorithms for MM, such as Hummel (2006); Krumm et al. (2007); Newson and Krumm (2009); Pink and Hummel (2010); Ren and Karimi (2009). The main differences between algorithms based on Viterbi lie in how measurement densities, transition probabilities and initial state distribution are estimated. A central property in the present paper is the general additivity described by

$$P[Y_i \in q, \Theta = \theta] = P[Y_i \in q', \Theta = \theta] + P[Y_i \in q'', \Theta = \theta],$$

(1)

where $Y_i$ is the actual position at time $t_i$, $q$ is a road segment, $\{q', q''\}$ is any partition of $q$ and $\Theta$ is anything, such as several measurements, locations at some points in time, etc. Note that (1) is an instance of Kolmogorov’s third axiom,

$$P(B_1 \cup B_2 \cup \ldots) = P(B_1) + P(B_2) + \ldots,$$

where $B_1, B_2, \ldots$ are mutually exclusive sets or events. We determine measurement densities in a way which satisfies (1) for the vast majority of possible partitions. We have failed to keep (1) satisfied in cases where the actual positions for two consecutive measurements lie on the same road segment and this segment is partitioned such that the positions end up on distinct parts, see Section 4.3 and 4.4. Transition probabilities and the initial state distribution are determined in a way which satisfies (1), see Section 4.5. The present author believes this to be the first paper where it is made explicit that (1) ought to be satisfied. It is not uncommon to use a function of the shortest distance between a measurement and a road to determine measurement probabilities Lou et al. (2009); Krumm et al. (2007); Newson and Krumm (2009); Ren and Karimi (2009), even though this contradicts (1), see Section 4.6.

The algorithm of the present paper also differs from previous algorithms in that it is based on a second order Hidden Markov Model (HMM), rather than a first order HMM as is common Hummel (2006); Krumm et al. (2007); Newson and Krumm (2009); Pink and Hummel (2010); Ren and Karimi (2009). This has two advantages and one disadvantage. The first advantage is that a first order HMM leads to treating noise as though it is white, while a second order HMM allows taking some account of correlation in noise. The second advantage is that transition probabilities are based on a longer history,
yielding much better possibilities for determining how likely the different transitions are. The disadvantage of a second order HMM as compared to a first order HMM is that the complexity of the algorithm increases with respect to several parameters. However, if the number of measurements is $n$ the complexity is still $O(n)$.

No attempts to use measurements of anything but time and horizontal positions have been made. Temporal information has been used only to a very small extent, in spite that earlier work indicates how heavy usage of temporal information may improve matching Lou et al. (2009). Roads are treated as polylines, in spite that earlier work indicates how treating roads as cubic splines may improve matching Pink and Hummel (2010). This ignorance is purposeful, in that it makes the theoretical contributions mentioned above as apprehensible as possible, not letting those contributions be obscured by too much detail of implementation.

In Section 2 all definitions and notations are gathered for ease of reference. Some related work is discussed in Section 3. Section 4 contains a mathematical description of MM, sets up the second order HMM and brings up how to determine the different types of densities and probabilities involved. In other words, this is where the results of the paper are contained. Section 5 contains an analysis of the complexity of using Viterbi on a second order HMM. In Section 6 it is shown that the algorithm obtains acceptable matchings, and Section 7 concludes the paper.

2. Definitions and Notation

A road is a directed curve from one point in space to another. A road network is a set of roads, which is connected, i.e. it is possible to move between any pair of roads by moving only along the roads in the network.

It is also good to have the underlying reality in mind. Roads are physical objects on the surface of the earth, and roads end or begin only at intersections. That roads end or begin only at intersections means that a road cannot be partitioned into several roads. Instead, we let each road consists of one or several road segments which we may partition further as we like.

The areas under consideration are typically small enough that a Euclidean geometry can be assumed. Also, our description of roads consider only one-way roads. This is not a restriction as a two-way road can be viewed as two one-way roads. When considering construction of any particular algorithm however, the possibility of U-turns anywhere along a two-way road should
be kept in mind. In the present work U-turns are dealt with in a somewhat off-hand fashion, see Section 4.5.

The task of MM is to find the route taken when given a road network and a series of measurements, i.e. positions with timestamps, generated from a journey along the roads in the road network. Here it is worthwhile to note that we assume knowledge of the road network. In reality, only maps are known, which are approximations of the road network. As soon as mathematical models are formed though, we can rarely do better than to assume that the maps are very similar to reality.

**Notation.** Points in time when measurements are made are denoted \( t_i, i \in \{1, \ldots, n\} \). Denote by \( \tau_i \) the amount of the time between \( t_{i-1} \) and \( t_i \), i.e. \( \tau_i = t_i - t_{i-1} \).

**Notation.** The measurement at \( t_i \) is denoted by \( z_i \), while the corresponding actual location is denoted by \( y_i \). The roads are denoted by \( r_i, i \in \{1, \ldots, N\} \), and the road network is denoted by \( \mathcal{R} \).

The starting point and the ending point of road \( r \) are denoted \( s(r) \) and \( e(r) \) respectively.

Measurements, actual positions, etc, are sometimes treated as random variables, in which case capital letters are used, e.g. \( z_i \) is the outcome of the random variable \( Z_i \).

**Definition.** For each \( i \in \{1, \ldots, N\} \), consider a partition of \( r_i \) into \( N_i \) road segments, such that we obtain the road segments

\[
q_j, j \in \{(1,1), \ldots, (1,N_1), \ldots, (N,1), \ldots, (N,N_N)\} = \mathcal{J}.
\]

Note that indices for road segments are pairs of integers. The starting point and the ending point of segment \( q \) are denoted \( s(q) \) and \( e(q) \) respectively.

Consider a state to be the segment at which the vehicle is located at time \( i \). When treating a state as random it is denoted \( Q_i \), and note that \( Q_i = q_j \) if and only if \( Y_i \in q_j \).

To maintain an obvious correspondence between indices, we use expressions such as \( Y_i \in q_{\phi(i)} \) where \( \phi : \{1, \ldots, n\} \mapsto \mathcal{J} \) is a function. We will use \( \phi_1 \) to denote the first element of \( \phi \), so that \( \phi_1 : \{1, \ldots, n\} \mapsto \{1, \ldots, N\} \), and \( r_{\phi_1(i)} \) is the road containing \( q_{\phi(i)} \).

**Notation.** Vectors of consecutive measurements, actual positions, states, etc, are denoted in bold, using the index to indicate which the first and the
last elements are, e.g. $Z_{i:j} = (Z_i, \ldots, Z_j)$ or $y_{i:j} = (y_i, \ldots, y_j)$, where $j \geq i$.

Sometimes a function of the index is wanted, e.g. $q_{\phi(i):j} = (q_{\phi(i)}, \ldots, q_{\phi(j)})$.

**Notation.** Inside the probability function $P[\cdot]$ and the (probability) density function $f(\cdot)$ shorthand notation is used. We write $z_i$ rather than $Z_i = z_i$, $y_i$ rather than $Y_i = y_i$, $q_{\phi(i)}$ rather than $Q_i = q_{\phi(i)}$, $r_{\phi(i)}$ rather than $Y_i \in r_{\phi(i)}$ and use corresponding shorthand notation when vectors of random variables are involved. Shorthand notation will be used also for events conditioned on.

**Notation.** In cases where we need to work separately in directions of latitude and longitude of some position or some noise, we use “NS” or “EW” as superscripts, e.g. $z_{i:}^{NS}$ or $\epsilon_{i:}^{EW}$.

**Notation.** We denote the great circle distance (distance along a great circle on earth, more or less the same as Euclidean when restricted to staying within a small region) between any two points $x$ and $y$ by $d_{GC}(x, y)$. We denote the minimal great circle distance between the point $z$ and the road $r$ by $d_{GC}(z, r)$.

We denote signed distance in the north-south direction between any two points $x$ and $y$ by

$$d_{NS}(x, y) = \frac{(x^{NS} - y^{NS}) (x^{EW} - y^{EW})}{|x^{NS} - y^{NS}| |x^{EW} - y^{EW}|} d_{GC}(x, (x^{NS}, y^{EW})) .$$

We denote signed distance in the east-west direction between any two points $x$ and $y$ by $d_{EW}(x, y)$. The concept of distance in east-west direction is not as precise as that in the north-south direction, but we determine $d_{EW}(x, y)$ as

$$d_{EW}(x, y) = \frac{(x^{NS} - y^{NS}) (x^{EW} - y^{EW})}{2|x^{NS} - y^{NS}| |x^{EW} - y^{EW}|} \times$$

$$(d_{GC}(x, (x^{NS}, y^{EW})) + d_{GC}((y^{NS}, x^{EW}), y)) .$$

The map is assumed to be small and we will stay away from the poles. This allows us to assume $d_{EW}^2(x, y) + d_{NS}^2(x, y) = d_{GC}^2(x, y)$.

We denote the route distance from a point $y_j \in R$ to a point $y_{j'} \in R$, i.e. shortest distance along roads in the road network, by $d_R(y_j, y_{j'})$. Note that route distance is not actually a distance measure as it is not symmetric.

We denote the length of a road $r$ by $d_R(r)$, which is very often, but not always, the same as $d_R(s(r), e(r))$. Similarly, we denote the length of a segment $q$ by $d_R(q)$.  


Notation. Let $I\{\cdot\}$ be the indicator function.

Notation. Let $\propto$ mean “proportional to”.

Notation. Let $y_j \preceq y_{j'}$ mean that $y_j$ and $y_{j'}$ lie on the same road $r$, and that when moving along $r$ from $s(r)$ to $e(r)$ one encounters $y_j$ no later than one encounters $y_{j'}$.

3. Related Work

For previous research on the topic of MM in general Quddus et al. (2007) is rather extensive, so we see no purpose in digging deeply into the topic. There are however some recent works on MM algorithms based on HMM or using Viterbi which ought to be mentioned.

The algorithm presented in Newson and Krumm (2009) produces good matchings, based on using route distances to determine transition probabilities. The transition probability from $r_{\phi_1(i-1)}$ to $r_{\phi_1(i)}$ is determined as

$$P[r_{\phi_1(i)}|r_{\phi_1(i-1)}] = \frac{1}{\beta} \exp\left(-\frac{|d_{GC}(z_i, z_{i-1}) - d_R(x_{i,\phi_1(i)}, x_{i-1,\phi_1(i-1)})|}{\beta}\right),$$

for some $\beta > 0$, where $x_{i,\phi_1(i)}$ is the point on $r_{\phi_1(i)}$ closest to $z_i$. Determining the transition probabilities in this manner is supported by an empirical study of the distribution of

$$|d_{GC}(z_i, z_{i-1}) - d_R(x_{i,\phi_1(i)}, x_{i-1,\phi_1(i-1)})|.$$

The approach must be seen as heuristic, as in a HMM the transition probabilities are independent on measurement probabilities.

Another important contribution of Newson and Krumm (2009) is that maps, along with measurement data and true path are published, so that comparisons between algorithms can be made on the same data set.

A good description of how to model with a first order HMM for the purpose of MM can be found in Ren and Karimi (2009), which deals with MM for wheelchair navigation. Wheelchair navigation differs quite a bit from that of car navigation. On one hand it is more difficult as sidewalks are located about twice as densely as roads and as “driving behaviour” is rather more irregular. Especially off-road travel, such as crossing a road or going through a park or a parking lot, is relatively common. On the other
hand, one doesn’t travel far between measurements, which makes it fairly inexpensive to find possible transitions. In Pink and Hummel (2010) we see an interesting approach, approximating the roads in the network not by straight lines as is common, but by cubic spline polynomials. Improvements in the MM procedure are reached by combining the improved interpretation of the map with inclusion of heading in the measurements. A first order HMM is set up and Viterbi is used to calculate a route.

The paper by Lou et al. (2009) focuses on MM where measurements are very sparse, and forms a clear indication as to the importance of using temporal information to improve MM in these situations. The paper makes no claim of using a HMM or of using Viterbi. Yet, one reason for considering the research Lou et al. (2009) as related to the research of the present paper is that the ST-algorithm presented in Lou et al. (2009) appears to be the well known Viterbi algorithm. The present author considers Lou et al. (2009) a strong indication that HMM for MM can benefit from utilization of the temporal information available, at least when measurements are sparse.

4. Hidden Markov Model

4.1. Choice of Model

The model we intend to use can be expressed in equational form as:

\[
\begin{align*}
\epsilon_i &\in N(0, \sigma) \times N(0, \sigma), \\
\xi_i &= \rho^{\tau_i} \xi_{i-1} + \sqrt{1 - \rho^{2\tau_i}} \epsilon_i, \\
Q_i &\in g(Q_{i-2}, Q_{i-1}, \tau_{i-1}, \tau_i), \\
Z_i &= \int_{Q_i} f(y|Q_i) y dy + \xi_i,
\end{align*}
\]

where \(C(\epsilon_i, \epsilon_j) = 0\) for \(i \neq j\) and \(g(\cdot, \cdot)\) is a function which defines some statistical distribution. The function \(g(\cdot, \cdot)\) is discussed further in Section 4.5. The dependencies in the model are depicted in Figure 1. In VanDiggelen (2007) it is shown that the noise in the plane is well approximated by circular bivariate Gaussian noise, and that it is not white. It is also claimed that short-term autocorrelation is positive, even though no model for an autocorrelation function is given. A larger and somewhat more recent study of noise is Wang et al. (2011), but that study doesn’t comment the correlation.
Figure 1: A second order HMM, where noise follows an AR(1)-process.
of noise at nearby points in time. The model for noise in the present paper is chosen to meet the claims of VanDiggelen (2007), while keeping algorithm complexity on an acceptable level. In particular, the marginal distribution of noise is Gaussian, \( \epsilon_{j}^{\text{NS}}, \epsilon_{j}^{\text{EW}} \in N(0, \sigma) \times N(0, \sigma) \).

The assumptions of the Markov model – which will be needed – are

\[
\begin{align*}
 f(z_{i}|z_{1:i−1}, q_{\phi(1:i)}) &= f(z_{i}|z_{1:i−1}, q_{\phi(i−1:i)}), \\
P[q_{\phi(i)}|q_{\phi(1:i−1)}] &= P[q_{\phi(i−2:i−1)}].
\end{align*}
\]

The distributions of \( (Y_{i−j:i}|Q_{i−j:i} = q_{\phi(i−j:i)}) \) will be needed to determine both measurement densities and transition probabilities. However, we have encountered difficulties in finding distributions of \( (Y_{i−j:i}|Q_{i−j:i} = q_{\phi(i−j:i)}) \) which both match common sense and conform with (1), (2) and (3). The distributions of \( (Y_{i−j:i}|Q_{i−j:i} = q_{\phi(i−j:i)}) \) are discussed in Section 4.3, where some heuristics for these distributions are determined.

4.2. The Objective

The objective is to calculate the route \( q_{\phi(1:n)} \) with the highest likelihood, given the measurements \( z_{1:n} \). Basically, we wish to maximize

\[
L = P[q_{\phi(1:n)}|z_{1:n}],
\]

with respect to \( \phi \). It is implicitly assumed that \( R \) and \( J \) as well as \( t_{i}, i \in \{1, \ldots, n\} \) are known, and conditioned on everywhere. Note that unless \( \tau_{i} \) is small for all \( i \), the route resulting from maximizing \( L \) with respect to \( \phi \) may contain gaps. We fill in the gaps in the route found by the shortest route. It is fully possible that there are better ways to fill in such gaps, e.g. prioritizing routes which contain only a few intersections or roads with high speed limits, but we anticipate that shortest route will be sufficient as long as data is not sampled very sparsely. Rewrite \( L \) as

\[
L = \frac{f(z_{1:n}|q_{\phi(1:n)})P[q_{\phi(1:n)}]}{f(z_{1:n})},
\]

and note that the denominator is independent of \( \phi \), and may without loss of generality be replaced by any constant, such as 1. Rewrite \( L \) once more, this time as

\[
L \propto \prod_{i=1}^{n} f(z_{i}|z_{1:i−1}, q_{\phi(1:i)})P[q_{\phi(i)}|q_{\phi(1:i−1)}].
\]
Using (2) and (3)

\[ L \propto f(z_{1:2}|q_{\phi(1:2)})P[q_{\phi(1:2)}] \prod_{i=3}^{n} f(z_i|z_{i-1}, q_{\phi(i-1:i)})P[q_{\phi(i)}|q_{\phi(i-2:i-1)}] \]

is obtained. While equivalent from a mathematical perspective, to avoid problems with the numerical precision of it is preferable to minimize \(-\log L\) rather than to maximize \(L\). It is common to call \(f(z_i|z_{i-1}, q_{\phi(i-1:i)})\) measurement, emission or observation densities, while \(P[q_{\phi(i)}|q_{\phi(i-2:i-1)}]\) are called transmission or transition probabilities. The probability \(P[q_{\phi(1:2)}]\) is called the initial state distribution. Note that while \(f(z_{1:2}|q_{\phi(1:2)})\) falls outside our naming conventions, how to calculate such expressions is covered en passant in Section 4.4.

4.3. Distribution of the Vehicle on a Given Road Segment

In this section the distribution of \((Y_i:j|Y_i:j \in q_{\phi(i:j)})\) is discussed. A natural assumption is that it has uniform marginal distributions, i.e.

\[ f(y_k|q_{\phi(k)}) = \begin{cases} d_R^{-1}(q_{\phi(k)}), & y_k \in q_{\phi(k)}, \\ 0, & \text{otherwise}, \end{cases} \quad (4) \]

for any \(k\). This is a slight approximation, as vehicles tend to keep somewhat lower speed in curves than they do on straight segments, and that a vehicle will typically have lower speed the closer it is to an intersection. Hence, density ought to be somewhat higher near intersections and in curves, but we believe these differences to be very slight and make this approximation without further ado.

It is not sufficient with (4) to calculate measurement densities and transition probabilities. To calculate measurement densities \(f(y_{i-1:i}|q_{\phi(i-1:i)})\) and \(f(y_{i-1}|q_{\phi(i-1:i)})\) are needed, see Section 4.4. To calculate transition probabilities we need \(f(y_{i-2:i}|q_{\phi(i-2:i)})\), see Section 4.5. Consider first cases where \(\phi(k) \neq \phi(k-1)\) for all \(k \in \{i-j+1, \ldots, i\}\). We are confident that \(f(y_{i-j:i}|q_{\phi(i-j:i)})\) is continuous on \(q_{\phi(i-j:i)}\). Hence, if segments are short, a good approximation which doesn’t contradict (4) is

\[ f(y_{i-j:i}|q_{\phi(i-j:i)}) = \begin{cases} \prod_{l=i-j}^{i} d_R^{-1}(q_{\phi(l)}), & y_{i-j:i} \in q_{\phi(i-j:i)}, \\ 0, & \text{otherwise}. \end{cases} \quad (5) \]

It is straightforward to show that both (4) and (5) conforms with (1) if and only if \(P[q]/d_{GC}(q) = P[q']/d_{GC}(q')\) holds for all road segments \(q\) and all partitions \(\{q', q''\}\) of \(q\), hence we assume that it holds.
Next, consider the case $j = 1$ with $\phi(i-1) = \phi(i)$. On one hand, common sense tells us that we are almost certain that $y_{i-1} \leq ... \leq x \leq d_R(s(q_{\phi(i)}), Y_i)$.

$$f(y_i | q_{\phi(i)}) = \sum_{j \in J} f(y_i, Y_{i-1} \in q_j | q_{\phi(i)}) = \sum_{j \in J} \int_{q_j} f(y_{i-1} | Y_{i-1} \in q_j, q_{\phi(i)}) dy_{i-1} =$$

$$= \sum_{j \in J} P[Y_{i-1} \in q_j] \int_{q_j} f(y_{i-1} | Y_{i-1} \in q_j, q_{\phi(i)}) dy_{i-1} =$$

$$= P[q_{\phi(i-1)}] \int_{q_{\phi(i-1)}} f(y_{i-1} | q_{\phi(i-1:i)}) dy_{i-1} +$$

$$+ \sum_{j \neq \phi(i-1)} P[Y_{i-1} \in q_j] \int_{q_j} f(y_{i-1} | Y_{i-1} \in q_j, q_{\phi(i)}) dy_{i-1} =$$

$$= P[q_{\phi(i-1)}] \int_{q_{\phi(i-1)}} f(y_{i-1} | q_{\phi(i-1:i)}) dy_{i-1} +$$

$$+ P[Y_{i-1} \notin q_{\phi(i-1)}] d_R^{-1}(q_{\phi(i)}) \mathbf{1}\{y_i \in q_{\phi(i)}\}. $$

But (4) yields

$$f(y_i | q_{\phi(i)}) = (P[q_{\phi(i-1)}] + P[Y_{i-1} \notin q_{\phi(i-1)}]) d_R^{-1}(q_{\phi(i)}) \mathbf{1}\{y_i \in q_{\phi(i)}\},$$

so with the highly reasonable assumption $P[q_{\phi(i-1)}] \neq 0$, we get

$$\int_{q_{\phi(i-1)}} f(y_{i-1} | q_{\phi(i-1:i)}) dy_{i-1} = d_R^{-1}(q_{\phi(i)}) \mathbf{1}\{y_i \in q_{\phi(i)}\};$$

or equivalently

$$f(y_i | q_{\phi(i-1:i)}) = d_R^{-1}(q_{\phi(i)}) \mathbf{1}\{y_i \in q_{\phi(i)}\}.$$

But similar calculations show

$$f(y_{i-1} | q_{\phi(i-1:i)}) = d_R^{-1}(q_{\phi(i-1)}) \mathbf{1}\{y_{i-1} \in q_{\phi(i-1)}\}.$$ 

As we are in the case $\phi(i-1) = \phi(i)$, we obtain

$$P(d_R(s(q_{\phi(i)}), Y_{i-1}) \leq x \leq d_R(s(q_{\phi(i)}), Y_i) | q_{\phi(i-1:i)}) =$$

$$= P(d_R(s(q_{\phi(i)}), Y_i) \leq x \leq d_R(s(q_{\phi(i)}), Y_{i-1}) | q_{\phi(i-1:i)}) = (6)$$
for all \( x \). The only way to satisfy both (6) and \( y_{i-1} \preceq y_i \) would be by assuming that \( \phi(i-1) = \phi(i) \) implies \( y_{i-1} = y_i \). We do not wish to make such a strong assumption, hence we must choose between the raw calculations leading to (6) and common sense which dictates \( \phi(i-1) = \phi(i) \). We believe it best to follow our common sense rather than our raw calculations, as the base for the above calculations are assumptions. Similar problems arise in the cases \( f(y_{i-1} | q_{\phi(i-1;i)}) \) with \( \phi(i-1) = \phi(i) \) and \( f(y_{1-2;i} | q_{\phi(i-2;i)}) \) with \( \phi(i-2) = \phi(i-1) \) or \( \phi(i-1) = \phi(i) \). In all of these cases we therefore intend to use heuristics rather than what could be calculated from our previous assumptions.

Heuristically, we choose to let

\[
f(y_{i-1} | q_{\phi(i-1;i)}) = 2d_R^2(q_{\phi(i)})I\{y_{i-1} \leq y_i\}I\{y_{i-1;i} \in q_{\phi(i-1;i)}\},
\]

when \( \phi(i-1) = \phi(i) \). This corresponds to let two independently and uniformly distributed points on \( q_{\phi(i)} \) be \( y_{i-1} \) and \( y_i \) in such a manner that \( y_{i-1} \preceq y_i \). Note that (7) in conjunction with (5) doesn’t conform with (1), which is straightforward by considering a road segment containing two consecutive positions, and a partition of the road segment such that the positions wind up on different parts. Similar problems exist for the heuristics below.

In accordance with (7), we obtain

\[
f(y_{i-1} | q_{\phi(i-1;i)}) = 2d_R(y_{i-1}, c(q_{\phi(i)}))d_R^2(q_{\phi(i)})I\{y_{i-1} \leq y_i\}I\{y_{i-1;i} \in q_{\phi(i-1;i)}\},
\]

when \( \phi(i-1) = \phi(i) \). Note that

\[
f(y_{i-1} | q_{\phi(i-1;i)}) = f(y_{i-1} | q_{\phi(i-1)}),
\]

otherwise.

We let \( f(y_{i-2;i} | q_{\phi(i-2;i)}) = f(y_{i-2;i-1} | q_{\phi(i-2;i-1)})f(y_i | q_{\phi(i)}) \) when \( \phi(i-1) \neq \phi(i) \) and \( f(y_{i-2;i} | q_{\phi(i-2;i-1)}) = f(y_{i-2-i} | q_{\phi(i-2)})f(y_{i-1;i} | q_{\phi(i-1;i)}) \) when \( \phi(i-2) \neq \phi(i-1) \). Based on how \( f(y_{i-2;i} | q_{\phi(i-2;i)}) \) is used in Section 4.5, we don’t need to know it explicitly in the case \( \phi(i-2) = \phi(i-1) = \phi(i) \). For our purposes it is enough to let

\[
y_{i-2} \preceq y_{i-1} \preceq y_i
\]

almost surely.

Note that in situation where \( \phi(i-1) = \phi(i) \) we violate some rules of probability theory. The solution we advocate is to place a sufficiently small limit on the length of segments. In the present paper, segments of length no
more than $\eta$ will be used. When $\eta$ is small situations where $\phi(i - 1) = \phi(i)$ will rarely be relevant. It is also worth noting that the suggested distributions of $(Y_{i-j;i} | Q_{i-j;i} = q_{\phi(i-j;i)})$ do not in general conform with (2) and (3), but they do conform if $\eta$ approaches 0. Limiting the lengths of segments has two disadvantages:

1. To obtain the probability that a vehicle has traversed a certain road, the probability that the vehicle was on the different segments of the road at a specific time should be summed up. Using Viterbi means that a maximum over the different segments is taken instead. Hence the algorithm can be expected to show an unwanted preference for suggesting a route with many short roads rather than a route with a few long roads, when operating on road segments rather than roads.

2. Using short segments increases memory requirements somewhat, and more importantly, it increases the complexity of Viterbi. The effect that limiting the length of segments has on the complexity of Viterbi is covered in Section 5.

4.4. Measurement Densities

The measurement probabilities can be rewritten as

$$f(z_i | z_{i-1}, q_{\phi(i-1;i)}) = \frac{f(z_{i-1} | q_{\phi(i-1;i)})}{f(z_{i-1} | q_{\phi(i-1;i)})}.$$  (8)

It is a simple matter to evaluate the numerator and the denominator. We need that for an $m$-variate Gaussian random variable $Z \in N(\mu, \Lambda)$ we have

$$f_x(x) = (2\pi)^{-m/2}(\det \Lambda)^{-1/2}\exp\left(-\frac{1}{2}(x - \mu)'\Lambda^{-1}(x - \mu)\right),$$

where $\Lambda$ is the covariance matrix.

We note that $(Z_j | Y_j = y_j)$ follows Gaussian distribution with

$$\Lambda = \begin{pmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{pmatrix}.$$
This results in
\[ f(z_{i-1}|q_{\phi(i-1:i)}) = \int_{q_{\phi(i-1)}} f(y_{i-1}|q_{\phi(i-1:i)}) \frac{1}{2\pi\sigma^2} \times \exp\left(-\frac{d_{NS}^2(z_{i-1}, y_{i-1})}{2\sigma^2} - \frac{d_{EW}^2(z_{i-1}, y_{i-1})}{2\sigma^2}\right) dy_{i-1} \] (9)

We note that \((Z_{i-1:i}|Y_{i-1:i} = y_{i-1:i}) \in N(y_{i-1:i}, \Lambda)\), where
\[
\Lambda = \sigma^2 \begin{pmatrix} 1 & 0 & \rho^{\tau_i} & 0 \\ 0 & 1 & 0 & \rho^{\tau_i} \\ \rho^{\tau_i} & 0 & 1 & 0 \\ 0 & \rho^{\tau_i} & 0 & 1 \end{pmatrix}
\]
for a certain order of the elements of \(Z_{i-1:i}\). We get
\[
f(z_{i-1:i}|q_{\phi(i-1:i)}) = \int_{q_{\phi(i-1)}} \int_{q_{\phi(i)}} \frac{f(y_{i-1}|q_{\phi(i-1:i)})}{4\pi^2\sigma^4(1 - \rho^{2\tau_i})} \times \\
\times \exp\left(-\frac{d_{GC}^2(z_{i-1}, y_{i-1}) + d_{GC}^2(z_i, y_i)}{2(1 - \rho^{2\tau_i})\sigma^2} + 2\rho^{\tau_i} \frac{d_{NS}(z_{i-1}, y_{i-1})d_{NS}(z_i, y_i)}{2(1 - \rho^{2\tau_i})\sigma^2} + \right. \\
\left. + 2\rho^{\tau_i} \frac{d_{EW}(z_{i-1}, y_{i-1})d_{EW}(z_i, y_i)}{2(1 - \rho^{2\tau_i})\sigma^2}\right) dy_i dy_{i-1},
\]
where some \(d_{NS}^2\) and \(d_{EW}^2\) terms have been gathered up into \(d_{GC}^2\) terms.

Now (8) can be evaluated without any problems, even if it has to be done numerically. There are however no real problems in numerical approximations of the integrals as the integrands are continuous, finite and have finite derivatives in the area of integration.

Note that as (8), (9) and (10) follows directly from laws of probability theory, our formula for \(f(z_i|z_{i-1}, q_{\phi(i-1:i)})\) conforms with (1) whenever our estimate of the distribution for \((Y_{i-j:i}|Q_{i-j:i} = q_{\phi(i-j:i)})\) does, i.e. whenever \(\phi(i - 1) \neq \phi(i)\).
For purposes of complexity of Viterbi it will be practical to approximate the measurement probability \( f(z_i|z_{i-1}, r_{\phi(i-1:i)}) \) by zero whenever \( d_{GC}(z_i, r_{\phi(i)}) \) is greater than some large constant distance. Call this constant distance \( \delta \). Similarly, for the first step, \( f(z_{1:2}|r_{\phi(1)}(i-1:i)) \) will be approximated by zero whenever \( \max(d_{GC}(z_{i-1}, r_{\phi(i-1)}), d_{GC}(z_i, r_{\phi(i)})) > \delta \). The effect \( \delta \) has on complexity is discussed in Section 5.

4.5. Joint Probability Distribution of States

Even though joint probability distributions of states do not appear explicitly in the solution to our optimization problem, it would be very interesting to be able to calculate them. The reason is that both the initial state distribution and the transition probabilities can be calculated directly from the joint probabilities of trigrams of states. Assume that \( P[q_{\phi(i-2:i-1)}] \), or at least a good approximation of probabilities of this form was known. We have

\[
P[q_{\phi(i-2;i-1)}] = \sum_{j \in \mathcal{J}} P[(q_{\phi(i-2)}, q_{\phi(i-1)}, q_j)],
\]

which for \( i = 3 \) is the initial state distribution. The transition probabilities are determined as

\[
P[q_{\phi(i)}|q_{\phi(i-2;i-1)}] = \frac{P[q_{\phi(i-2;i)}]}{P[q_{\phi(i-2;i-1)}]}.
\]

One way to conform to (1) is to estimate the joint probabilities of trigrams of states by

\[
P[q_{\phi(i-2;i)}] \propto \int_{q_{\phi(i-2)}} \int_{q_{\phi(i-1)}} \int_{q_{\phi(i)}} f(y_{i-2;i}, r_{\phi_1(i-2;i)}) dy_i dy_{i-1} dy_{i-2},
\]

for some well-behaved nonnegative function \( f \). While any well-behaved nonnegative function \( f \) will suffice for (1), \( f \) should really be the joint density function for \((y_{i-2;i}, r_{\phi_1(i-2;i)})\). Note that using just \( f(y_{i-2;i}) \) as integrand is insufficient as the direction would often be undetermined. The density function is of course unknown, but it can be approximated by a function corresponding to reasonable driving behaviour.

It is fairly reasonable driving behaviour to take the shortest route between the two points one wishes to travel between. Hence, \( f \) should favour short routes over long routes. For the purpose of estimating \( P[q_{\phi(i-2;i)}] \), the driver
has almost certainly attempted to travel from $s(r_{\phi_1(i-2)})$ to $e(r_{\phi_1(i)})$, where
the exceptions are the start of a journey, the end of the journey or include
at least one U-turn on $r_{\phi_1(i-2)}$ or $r_{\phi_1(i)}$. To simplify calculations we proceed
as though starts of journeys, end of journeys and U-turns can only occur at
intersections. This treatment of U-turns is almost certainly suboptimal, but
it is sufficient to illustrate the theoretical novelties of this paper.

It is also reasonable that one must have enough time to travel between
the necessary roads, which is checked by an indicator function permitting
speeds less than or equal to some constant $\nu$. A function depending on the
speed limits of the roads on the shortest route and dropping to zero in a
more continuous fashion may give better results, but the indicator function
leads to fairly simple integration and is sufficient to illustrate the theoretical
novelties of this paper. Note that for the purpose of algorithm complexity is
important to use a function which is zero for all but some easily identifiable
roads. The effect $\nu$ has on algorithm complexity is covered in Section 5.

In the algorithm in the present paper,

- the function used is

$$f(y_{i-2:i}, r_{\phi_1(i-2:i)}) = \mathbf{1}\{d_R(y_{i-2}, y_{i-1}) \leq \nu\tau_{i-1} \cap d_R(y_{i-1}, y_i) \leq \nu\tau_i\}, \quad (12)$$

when $\phi_1(i-2) = \phi_1(i-1) = \phi_1(i)$ and $y_{i-2} \preceq y_{i-1} \preceq y_i$,

- the function used is

$$f(y_{i-2:i}, r_{\phi_1(i-2:i)}) = \mathbf{1}\{d_R(y_{i-2}, y_{i-1}) \leq \nu\tau_{i-1} \cap d_R(y_{i-1}, y_i) \leq \nu\tau_i\} \times
\exp \left(\gamma R\left(d_R(s(r_{\phi_1(i-2)}), e(r_{\phi_1(i)})) - d_R(r_{\phi_1(i-2)}) -
\begin{array}{c}
d_R(e(r_{\phi_1(i-2)}), s(r_{\phi_1(i)})) - d_R(r_{\phi_1(i)})
\end{array}/(\tau_{i-1} + \tau_i)\right)\), \quad (13)$$

when

- $\phi_1(i-2) = \phi_1(i-1) = \phi_1(i)$ and $y_{i-2} \preceq y_i \preceq y_{i-1}$,
- $\phi_1(i-2) = \phi_1(i-1) = \phi_1(i)$ and $y_{i-1} \preceq y_{i-2} \preceq y_i$,
- $\phi_1(i-2) = \phi_1(i-1) = \phi_1(i)$ and $y_{i-1} \preceq y_i \preceq y_{i-2}$,
- $\phi_1(i-2) = \phi_1(i-1) = \phi_1(i)$ and $y_i \preceq y_{i-2} \preceq y_{i-1}$,
\[ \phi_1(i-2) = \phi_1(i-1) \neq \phi_1(i) \text{ and } y_{i-2} \preceq y_{i-1} \text{ or} \]
\[- \phi_1(i-2) \neq \phi_1(i-1) = \phi_1(i) \text{ and } y_{i-1} \preceq y_i, \]

- the function used is

\[ f(y_{i-2:i}, \phi_1(i-2:i)) = \]
\[ \mathbf{1}\{d_R(y_{i-2}, y_{i-1}) \leq \nu \tau_{i-1} \cap d_R(y_{i-1}, y_i) \leq \nu \tau_i\} \times \]
\[ \exp \left( \gamma_R (d_R(s(r_{\phi_1(i-2):e(r_{\phi_1(i)}))) - d_R(r_{\phi_1(i-2):s(r_{\phi_1(i-1)})) - d_R(s(r_{\phi_1(i-1):r_{\phi_1(i)}}))) / (\tau_{i-1} + \tau_i) \right) \]

in all other cases,

where \( \gamma_R > 0 \) is some constant representing how unlikely we believe it is for a vehicle to avoid the shortest route.

4.6. The Impact of Kolmogorov’s Third Axiom

This section considers the impact of (1) on MM. Analysis is performed with respect to first order HMM, which makes it applicable to earlier work.

First, it is shown that determining measurement densities – \( f(z_i|q_{\phi(i)}) \) – by a nonincreasing function of \( d_{GC}(z_i, q_{\phi(i)}) \) is an insufficient approach to satisfy (1). Note that such an approach has been taken on several occasions Lou et al. (2009); Krumm et al. (2007); Newson and Krumm (2009); Ren and Karimi (2009). To aid our arguments, we will use Figure 2, in which we see a measurement, \( z_i \), and a partition \( \{q'_{\phi(i)}, q''_{\phi(i)}\} \) of the road segment \( q_{\phi(i)} \). According to (1) we should have
\[
f(q_{\phi(i)}, z_i) = f(q'_{\phi(i)}, z_i) + f(q''_{\phi(i)}, z_i), \tag{15}
\]
which is equivalent to
\[
f(z_i|q_{\phi(i)})P[q_{\phi(i)}] = \ldots > d GC(x_i, \phi(i), e(q'_{\phi(i)})).
\]

Determining \( f(z_i|q_{\phi(i)}) \) by a function of \( d_{GC}(z_i, q_{\phi(i)}) \) would yield \( f(z_i|q_{\phi(i)}) = f(z_i|q'_{\phi(i)}) \), so we obtain
\[
f(z_i|q_{\phi(i)})(P[q_{\phi(i)}] - P[q'_{\phi(i)}]) = f(z_i|q''_{\phi(i)})P[q''_{\phi(i)}],
\]
but from (1) we obtain \( P[q_{\phi(i)}] - P[q'_{\phi(i)}] = P[q''_{\phi(i)}] \), so the absurd conclusion \( f(z_i|q_{\phi(i)}) = f(z_i|q''_{\phi(i)}) \) would follow.

While from a theoretical perspective it is obvious that (1) should be fulfilled, one may ask how important it is in practice. Are there any significant differences in determining \( f(z_i|q_{\phi(i)}) \) by a nonincreasing function of \( d_{GC}(z_i, q_{\phi(i)}) \), and by satisfying (1)? We show that when (1) is satisfied, then the measurement densities are approximately inversely proportional to the asymptotic probability of being at those road segments, at least for sufficiently long and straight road segments. This is a significant property which is not attained when determining \( f(z_i|q_{\phi(i)}) \) by a nonincreasing function of \( d_{GC}(z_i, q_{\phi(i)}) \).

Let us return to Figure 2. Assume that north is the direction upwards in the figure. Denote \( e(q''_{\phi(i)}) = d_{\phi(i)}' \cap q''_{\phi(i)} \). First we wish to show that \( f(z_i, q''_{\phi(i)})/f(z_i, q'_{\phi(i)}) \to 0 \) as \( d_{R}(q''_{\phi(i)}) \to \infty \), no matter how much faster \( d_{R}(q''_{\phi(i)}) \to \infty \). Let \( x_{i,\phi(i)} \) be the point on \( q_{\phi(i)} \) closest to \( z_i \), where \( x_{i,\phi(i)} \) divides \( q_{\phi(i)}'' \) in two pieces with some fixed ratio so that \( d_{GC}(x_{i,\phi(i)}, e(q'_{\phi(i)})) \to \infty \) as \( d_{R}(q''_{\phi(i)}) \to \infty \). We have
\[
\frac{f(z_i, q''_{\phi(i)})}{f(z_i, q'_{\phi(i)})} = f(\xi_i^{NS} = d_{GC}(z_i, q_{\phi(i)})) \int_{q''_{\phi(i)}} f(\xi_i^{EW} = d_{GC}(x_i, y_i))dy_i
\]
\[
\frac{f(z_i, q'^{NS}_{\phi(i)})}{f(z_i, q'_{\phi(i)})} \int_{q'_{\phi(i)}} f(\xi_i^{EW} = d_{GC}(x_i, y_i))dy_i,
\]
but
\[
\int_{q''_{\phi(i)}} f(\xi_i^{EW} = d_{GC}(x_i, y_i))dy_i < P(\xi_i^{EW} > d_{GC}(x_{i,\phi(i)}, e(q'_{\phi(i)})))
\]
and
\[
\int_{q''_{\phi(i)}} f(\xi_i^{EW} = d_{GC}(x_i, y_i))dy_i > P(0 < \xi_i^{EW} < d_{GC}(x_{i,\phi(i)}, e(q'_{\phi(i)}))).
\]
We thus have
\[
\frac{f(z_i, q''_{\phi(i)})}{f(z_i, q'_{\phi(i)})} < \frac{P(e_{\text{EW}}^i > d_{GC}(x_i, \phi(i), e(q'_{\phi(i)})))}{P(0 < e_{\text{EW}}^i < d_{GC}(x_i, \phi(i), e(q'_{\phi(i)})))} \rightarrow 0
\]
as \(d_{GC}(x_i, \phi(i), e(q'_{\phi(i)})) \rightarrow \infty\). But from (1) we had (15) which we may rewrite as
\[
\frac{f(z_i, q_{\phi(i)})}{f(z_i, q'_{\phi(i)})} = 1 + \frac{f(z_i, q''_{\phi(i)})}{f(z_i, q'_{\phi(i)})}.
\]
For sufficiently long \(q'_{\phi(i)}\) we have
\[
\frac{f(z_i, q_{\phi(i)})}{f(z_i, q'_{\phi(i)})} \approx 1,
\]
or equivalently
\[
f(z_i | q_{\phi(i)}) P[q_{\phi(i)}] \approx f(z_i | q'_{\phi(i)}) P[q'_{\phi(i)}],
\]
which is what we wanted to show. It is reasonable that \(P[q_{\phi(i)}] \propto d_R(q_{\phi(i)})\), at least approximately. Lengths of roads frequently differ by a factor 100 or more, so when entire roads are used as road segments, this can be expected to have a large effect.

5. Complexity of Viterbi

The algorithm typically used when dealing with HMM is Viterbi. It has been used for MM earlier as well, see Section 3 for some examples of this, but as far as is known to the author of the present paper this is the first time it is used for a second order HMM for MM. Note that Viterbi has been used for second order HMM in other applications, such as part-of-speech recognition Thede and Harper (1999) and speech recognition Watson and Chung Tsoi (1992), to name a few. Before suggesting the use of Viterbi for a second order HMM for MM, the complexity must be considered.

The total number of states is \(|\mathcal{J}|\). Assuming that the time required to calculate one measurement probability or one transition probability is \(O(1)\), the complexity of Viterbi for a \(k\):th order HMM is then \(O(|\mathcal{J}|^{k+1}n)\). However, not all transitions are possible, and according to assumptions made, not all states are relevant to consider.

By the remark at the end of Section 4.4 the measurement probability to a state \(q_j\) at \(t_i\) is approximated by zero unless \(d_{GC}(z_i, q_j) \leq \delta\). The number
of states with nonzero measurement density is for reasonable $\delta$ much smaller than $|J|$, and is restricted by

$$\max_{v \in V} |\{j \in J : d_{GC}(v, q_j) \leq \delta\}| = O(\delta^2 / \eta),$$

where $V$ is the face of the earth. Assuming that the time required to calculate one measurement density and one transition probability is $O(1)$, the complexity of Viterbi for second order HMM is $O(n \delta^6 / \eta^3)$, as identifying the relevant states swiftly enough is easy when using an appropriate data structure.

Note that as the calculation of a measurement probability at time $t_i$ is independent of the state at $t_{i-2}$, only $O(n \delta^4 / \eta^2)$ measurement probabilities need to be calculated. Still, the complexity of Viterbi is at least $O(n \delta^6 / \eta^3)$. The complexity for calculating a measurement density as in (8) by numerical approximation is clearly independent with respect to $n$, $\delta$, $\nu$ and $t_i$, $i \in \{1, \ldots, n\}$. It may depend on $\eta$, as the area of integration is $O(\eta^2)$, but only in such a way that it is calculated faster for smaller $\eta$, so that we may without loss assume the complexity for calculating a measurement density to be constant.

The complexity for calculating a transition probability as in Section 4.5 is a far more difficult case. It is mainly the route distances which cause us worries. We assume route distances are calculated by Dijkstra’s algorithm Dijkstra (1959), henceforth referred to as Dijkstra. In particular, Dijkstra needs to be used only on the graph built from the roads, not on the graph built from the segments, as knowing route distance between a pair of roads allows for constant time calculation of route distances between segments on those roads. Note that Dijkstra calculates the shortest distances from one road to all other roads. The indicator function included in our estimates of joint density functions, see Section 4.5, allows us to end Dijkstra early. At the step of the algorithm corresponding to time $t_i$, the calculations of route distances are majorized by the calculation of route distances to all roads within $(\tau_i + \tau_{i+1}) \nu$ of every road within $\delta$ of $z_i$. This means finding distances to $O((\tau_i + \tau_{i+1})^2 \nu^2)$ roads from each of $O(\delta^2)$ roads. The complexity of Dijkstra is $O((v + e) \log(e))$, where $v$ is the number of vertices and $e$ is the number of edges in the graph. In the present graph, the degree of each node, i.e. the number of roads adjacent to a road, is guaranteed to be less than at least some small constant. Hence, $e = O(v)$ and Dijkstra has complexity $O((\tau_i + \tau_{i+1})^2 \nu^2 \log((\tau_i + \tau_{i+1})^2 \nu^2))$. The total amount of work required to
run Dijkstra within Viterbi is

\[ O \left( \frac{(\delta \nu)^2}{\eta} \sum_{i=1}^{n-2} (\tau_i + \tau_{i+1})^2 \log((\tau_i + \tau_{i+1})\nu) \right) . \]

To calculate the joint probability distribution of states according to (11), in the step of the algorithm corresponding to \( t_i \) there are \( O(\delta^4/\eta^2) \) sums, each summing \( O((\tau_i + \tau_{i+1})^2\nu^2/\eta) \) terms, leading to

\[ O \left( \frac{\delta^4 \nu^2}{\eta^3} \sum_{i=1}^{n-2} (\tau_i + \tau_{i+1})^2 \right) \]

operations being required.

The total complexity of running Viterbi for our second order HMM is

\[ O \left( \frac{\delta^2}{\eta} \left( \frac{n\delta^4}{\eta^2} + \nu^2 \sum_{i=1}^{n-2} (\tau_i + \tau_{i+1})^2 \left( \log((\tau_i + \tau_{i+1})\nu) + \frac{\delta^2}{\eta^2} \right) \right) \right) . \]

When \( \tau = \max_i \tau_i \) the complexity can be expressed as

\[ O \left( \frac{n\delta^2}{\eta} \left( \frac{\delta^4}{\eta^2} + (\nu \tau)^2 \left( \log(\nu \tau) + \frac{\delta^2}{\eta^2} \right) \right) \right) . \]

This second formula is a terrible estimate if the series of measurements is dense in general but contains a few large gaps.

It is relevant to note that the given algorithm fails to run in real-time if some \( \tau_i \) is too large. The occasional gap in data can be handled by starting a new route, but if one wants an algorithm which runs in real-time also in cases where \( \tau_i \) is large rather frequently, one needs to find a way of determining transition probabilities which doesn’t depend on route distances.

6. Test Results

The algorithm was tested on the data and map provided in Newson and Krumm (2009). We start by discussing some problems regarding the map and the data.

The map doesn’t provide reliable information regarding which road is incident to which road. Every road is tagged with a From Node ID and
a To Node ID, which ought to be the information needed. However, there are many cases where a road \( r_i \) is incident to a road \( r_j \) even though the To Node ID of \( r_i \) is distinct from the From Node ID of \( r_j \). On the other hand, there are many cases where \( e(r_i) = s(r_j) \) even though \( r_i \) is not incident to \( r_j \), which is due to viaducts or tunnels, and this is in the map data indicated only by the To Node ID of \( r_i \) being distinct from the From Node ID of \( r_j \).

In the present work, this has been handled by assuming that it is possible to drive from \( r_i \) to \( r_j \) without passing other roads whenever \( e(r_i) = s(r_j) \), unless the To Node ID of \( r_i \) is distinct from the From Node ID of \( r_j \) and at least one more road (not counting two-way roads twice) has an endpoint in \( e(r_i) \).

The route lacks certain features. For example, it includes no U-turns. This leads to that an algorithm’s propensity to suggest that a U-turn has been made is not properly evaluated.

The measurements are not really noisy enough to permit valuable comparisons on very dense data. In previous research Jawad and Kersting (2010); Newson and Krumm (2009) data has been degraded by subsampling and adding white noise. As we believe that noise is far from white, we do not consider it an option to add white noise as a means of degrading the data.

The parameters used for testing the algorithm were \( \rho = 0.95 \), \( \sigma = 7 \) m, \( \gamma_R = 1 \), \( \delta = 60 \) m, \( \nu = 40 \) m/s and \( \eta = 40 \) m. The values have not been optimized, but they were tested on a small set of private measurements and turned out to be sufficient to perform acceptably when data was sampled periodically with periods 10 s and 30 s. It would be preferable to use higher higher values for \( \delta \) and \( \nu \); the rather restrictive choices of these parameters were made with respect to complexity.

It should also be stated that in the implementation of Viterbi it was included that each time the algorithm moved to a new step in time, states with probabilities deemed less than \( 10^{-20} \) times as likely as the currently most likely state were dropped. This was made in order to speed up the algorithm somewhat. The author of the present work believes that this has had no effect on the performance of the algorithm, apart from the speed with which it has operated.
Table 1: Performance of Viterbi’s algorithm for a second order HMM.

<table>
<thead>
<tr>
<th>rate</th>
<th>mismatched fraction</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/10 s$^{-1}$</td>
<td>0.00%</td>
<td>1</td>
</tr>
<tr>
<td>1/20 s$^{-1}$</td>
<td>0.14%</td>
<td>0.999</td>
</tr>
<tr>
<td>1/30 s$^{-1}$</td>
<td>0.48%</td>
<td>0.998</td>
</tr>
<tr>
<td>1/45 s$^{-1}$</td>
<td>7.51%</td>
<td>0.962</td>
</tr>
<tr>
<td>1/60 s$^{-1}$</td>
<td>4.79%</td>
<td>0.976</td>
</tr>
<tr>
<td>1/120 s$^{-1}$</td>
<td>14.72%</td>
<td>0.926</td>
</tr>
</tbody>
</table>

The results of running the algorithm on the data from Newson and Krumm (2009) are displayed in Table 1. Some words on the measures of performance:

- The mismatched fraction is used in Jawad and Kersting (2010); Newson and Krumm (2009). Denote the length of the roads in the retrieved route which are not also in the ground truth by $L_p$. Denote the length of the roads in the ground truth which are not also in the retrieved route by $L_r$. Denote the length of the ground truth by $L_{gt}$. Then the mismatched fraction is measured as $(L_p + L_r)/L_{gt}$.

- The F-measure is a common way of measuring performance for information retrieval. Denote the precision (the fraction of the retrieved route which is part of the ground truth) by $p$. Denote the recall (the fraction of the ground truth which is part of the retrieved route) by $r$. The F-measure is determined as $2pr/(p + r)$.

- A performance measure reported in Quddus et al. (2007) is the percentage of correctly matched links. For this measure, a match is considered correct only if it is correct with respect to both time and location. As no timestamps are supplied for the ground truth of Newson and Krumm (2009), such a performance measure cannot be used here.

One thing that should be noted about the results in the table above is that the performance at rate $1/45$ s$^{-1}$ is much worse than at $1/60$ s$^{-1}$. Higher rate sometimes resulting in lower performance than lower rate is a phenomenon which appeared in Newson and Krumm (2009) as well. This shows that MM is rather sensitive to precisely which measurements are maintained when degrading data by reducing the rate. Not using identical sets
of measurements can be expected to account for quite large differences in reported performance of algorithms. Hence, it is important to report that the degradation of data in the present paper to achieve rate $\frac{1}{x} \text{s}^{-1}$ was made by maintaining measurements $1, 1+x, 1+2x, \ldots$. Gaps in the original data means that actual rate was slightly below what is reported in Table 1.

It is difficult to compare the results in a precise manner to those of Newson and Krumm (2009), where results are reported graphically rather than in numerical form, but the results appear to be similar.

Some suggestions for improvements of the presented algorithm are:

- The function used for distribution of position on a given road, discussed in Section 4.3, was shown to be somewhat incorrect. This may cause significant errors, especially when the vehicle remain on the same road segment for consecutive measurements. Fixing this issue may have a positive effect on the performance of the algorithm, especially when measurements are very dense.

- The function for estimating transition probabilities as in (12), (13) and (14) is only a guess. Most likely a function modelling reality better can be found.

- A large part of the errors for rate $1/120 \text{s}^{-1}$ are the result of using the shortest route to fill in gaps that are present after matching one road segment to every measurement. A better route planner may improve this step of the matching process.

- U-turns are treated in a sloppy manner. This produce a few matching errors, and the algorithm may be improved by treating the possibility of U-turns more seriously.

7. Conclusion

In the present paper some theoretical novelties for probabilistic MM algorithms have been presented.

First and foremost the use of second order HMM for MM has been presented. This permits using autocorrelation of noise in the matching process and to make better use of “recent history” of where one has been driving. It is reasonable to expect that the better model for behaviour of noise has an effect mainly for dense measurements, as noise is close to white when
measurements are sparse. It is reasonable to expect that “recent history” of where one has been driving has an effect mainly when measurements are sparse, as little information is gained from this when one stays on the same road for consecutive measurements.

Apart from working with a second order HMM instead of a first order HMM, a principle of general additivity, (1), and the effect this ought to have on measurement densities and transition probabilities have been discussed. Measurement densities have been determined in a manner nearly conforming with general additivity, and transition probabilities have been determined in a manner conforming fully with the general additivity. An algorithm based on these measurement densities and transition probabilities have been put together. On the one hand it doesn’t outperform state-of-the-art algorithms, but it performs well and has a lot of room for improvement.

Acknowledgements

The author would like to thank Mattias Dahl, Johan Holmgren, and Håkan Lennerstad (all three from BTH, Sweden) for some interesting discussions regarding the content of this paper.

References


Combinatorial optimization is a diverse area of mathematics. It concerns optimization on feasible regions defined by discrete sets, graphs, hypergraphs, matroids, etc... which all have a large number of applications. They occur in virtually all domains of human activity, since humans always want to do things easier, faster, consume less resources, etc... This thesis concerns three applied problems within combinatorial optimization.

The first paper generalizes previous optimal upper bounds on the minimum Euclidean distance for phase-shift keying (PSK) block codes, that are explicit in the parameters alphabet size, block length and code size. There is a strong connection between high minimum Euclidean distance and good error-correcting capabilities. The bounds are generalized in several respects, such as from codes on symmetric PSK to codes on asymmetric PSK. They are also generalized to other types of noise than Gaussian, allowing more efficient block codes when noise is non-Gaussian. We provide examples of codes on asymmetric PSK that have higher minimum Euclidean distance than any comparable codes on symmetric PSK. Several classes of codes are shown to be optimal among codes on symmetric PSK since their Euclidean distance coincides with the bound.

The second paper considers a parallel computer system with \( m \) identical computers, where we study optimal performance precaution for one possible computer crash. We anticipate that some computer may crash, and restrict the cost in such a situation. How costly is such a precaution when no crash occurs? We set a restriction that the completion time of a parallel program after a crash is at most a factor \( r + 1 \) larger than if we use an optimal allocation with \( m - 1 \) computers. This is an \( r \)-dependent restriction of the set of allocations of a program. Then the worst-case ratio of the optimal \( r \)-dependent completion time in the case of no crash and the unrestricted optimal completion time defines a function \( f(r,m) \). In the paper we establish upper and lower bounds of the worst-case cost function \( f(r,m) \) and characterize worst-case programs.

The third paper considers the problem of Map Matching (MM), i.e. matching time and location measurements of a vehicle to a route in a road network. The paper presents a probabilistic algorithm for MM based on a second order hidden Markov model (HMM), as opposed to first order HMMs which are usually used. This allows a more detailed analysis of the data while preserving algorithmic complexity \( O(n) \). Both measurement densities and transition probabilities are determined with respect to Kolmogorov’s third axiom, which in this context implies that the probabilities are additive over a partition of a road segment.