*Mathl. Comput. Modelling* Vol. 16, No. 6/7, pp. 113–119, 1992
Printed in Great Britain. 

0895-7177/92 $5.00 + 0.00


# PHAGE TYPING SETS

GEORGE MARKOWSKY
Computer Science Department
University of Maine, Orono, ME 04469, U.S.A.
MELVIN GERSHMAN
Microbiology Department
Animal and Veterinary Sciences Department
University of Maine, Orono, ME 04469, U.S.A.
JACQUELINE HUNTER *
Animal and Veterinary Sciences Department
University of Maine, Orono, ME 04469, U.S.A.

**Abstract** — Bacteria are susceptible to phage (viral) infections. One of the most important properties of phages is host specificity. Phages can be so selective that they will distinguish varieties among apparently identical organisms. By exposing an isolate growing on the surface of an agar plate to a battery of different phages a pattern develops contingent on the susceptibility or resistance of the culture to the phages. A culture sensitive to a particular phage is destroyed and the destruction is manifested by areas devoid of bacterial growth. Using phages to differentiate bacteria is referred to as phage typing.

Phage typing can be extremely important in many health situations because it can identify random, unrelated organisms as well as the isolates that are actually responsible for a given problem. Aside from relating an organism to an outbreak, this laboratory method can also be used for surveillance, assessing strain distribution, and ascertaining the effectiveness of therapeutic measures. Phage typing requires the use of a standard collection of dissimilar phages. In the process of developing a phage typing set, numerous phages are first isolated and tests are undertaken to determine if they are different and useful in delineating the types of organisms under study. To make the procedure more cost effective and less labor-intensive the final set is reduced to a manageable number with the assistance of a computer.

Selecting a smallest phage typing set is an $NP$-hard combinatorial search problem so no efficient algorithm for finding it is believed to exist. However, with a powerful computer very good solutions and sometimes optimal solutions can be found by intelligent searching.

Using a variety of heuristic techniques and a variety of computers small phage sets have been found previously for *Salmonella, Escherichia coli*, and *Staphylococcus epidermidis*. These programs have been rewritten to take advantage of the greater speed and address space of the IBM 3090. We are now able to obtain better results because searching can be done more exhaustively and are able to find optimal phage typing sets, something that we were not able to do with earlier programs running on less powerful computers. We are currently working on a phage typing set for *Staphylococcus aureus*.

Besides finding optimal solutions we also tested a greedy algorithm. Our results suggest that the greedy algorithm works well and should enable us to solve very large problems using powerful computers.

## 1. INTRODUCTION

Epidemiological testing requires fast, accurate and economical techniques. Phage typing is one such technique. Most of the raw data needed for phage testing must be gathered by lots of manual labor without computer assistance. The data that is collected has a random nature and is typically voluminous enough that it is difficult for humans to spot key patterns. The high data

*The authors would like to thank the Maine Agricultural Experiment Station for supporting the biological work, and the members of the University of Maine System Computing Center for their support of the programming efforts described in this paper. The authors would also like to thank Mary Preble for helping with some of the data entry.

processing speed of the computer can organize the data and make the final test as efficient and economical as possible.

Section 2 of this paper describes the importance of phage typing, briefly describes how the raw data for a phage typing set is collected, and explains why it is important to make a phage typing set as small as possible. Section 3 describes how the computer can be used to find small phage typing sets. Section 4 briefly describes $NP$-complete and $NP$-hard problems and the relevance of such problems to the problem of finding small phage typing sets. Section 5 describes the programs that we have developed to find small phage typing sets and shows why a high speed computer is a valuable asset in the search for small phage typing sets.

The authors believe that computers can be of great benefit to scientists working in the life and health sciences in many ways besides record keeping. We hope that this paper will inspire other researchers to make greater use of computers in their own work. In particular, the results of this work on phage typing can be applied directly to any other typing method which can be summarized by a table of hits and misses.

## 2. PHAGE TYPING

Phage typing is an extremely valuable diagnostic techniques. Our phages have been used to relate isolates to disease outbreaks in hospitals, processing plants, a campus and a restaurant. Furthermore, they have been requested and used in a number of countries throughout the world including Switzerland, Canada, Japan, Israel, Holland and the Center for Disease Control (CDC) in the United States. References [1–11] give an example of the wide range of applications of phage typing.

To create a good phage typing set you should begin with a good source of phage. Sewage samples are an excellent source of phage. Following appropriate incubation, sewage samples that were inoculated with bacteria are bacteriologically filtered and examined for phage. The presence of phage is detected by spotting cultures, used as the inoculum, with their respective filtrates. A culture reacting to the presence of phage will be destroyed and "plaques" will be observed.

Phage isolates, when detected, are purified, brought to a usable titre and tested against a number of other organisms of the same group to determine if they are suitable for making type differentiations and should be included in a standard set. For more details see the book by Adams [12].

The techniques just described are adequate for constructing a phage typing set. There are, however, several reasons for making the phage typing set as small as possible. Two of the key reasons are:

1. *Speed of the test is improved.* Clearly, with fewer phages to apply to bacterial cultures the speed of the test is improved.

2. *The testing becomes more economical.* With fewer strains needed, less money and effort is needed to keep the test viable.

Producing small phage typing sets is complicated by the fact that it is sometimes difficult to keep particular phage strains alive. Using the computer allows the researcher to quickly determine the impact on testing of the loss of particular phages so that a stable set of discriminating phages may be selected.

## 3. THE COMPUTER'S ROLE

To the computer, the raw phage data looks like a table with the names of the bacterial strains labeling the rows and the names of the phage strains labeling the columns. Each entry in the table is a 1 or 0 depending on whether or not the phage destroys that particular bacterial strain. The goal of the computer program is to find the smallest number of phages (columns) such that any two bacteria (rows) differ in at least one instance (entry).

Table 1 is of the form discussed in this paper. It contains just 5 strains of bacteria and 4 strains of phage. A 1 entry in the table means that the phage used to label the column killed the bacteria used to label the row. A 0 means that the phage used to label the column did not kill the bacteria used to label the row.

Table 1. A simple, sample data set.

| | Phage 1 | Phage 2 | Phage 3 | Phage 4 |
|---|---|---|---|---|
| Bacteria 1 | 0 | 1 | 1 | 0 |
| Bacteria 2 | 0 | 0 | 1 | 1 |
| Bacteria 3 | 0 | 0 | 0 | 1 |
| Bacteria 4 | 1 | 0 | 0 | 0 |
| Bacteria 5 | 1 | 1 | 0 | 0 |

It is clear that each strain of bacteria in Table 1 has a kill pattern that is different from every other strain in the table. Note, however, that not all phages are necessary to distinguish the strains from one another. In particular, if only the first 3 phages are used, all the strains still have a unique kill pattern. Thus, we can consider the set of phages {1,2,3} as equally effective for typing as {1,2,3,4}.

Further analysis of Table 1 shows that the set of phages {2,3,4} will also distinguish among the bacteria. Both {1,2,3} and {2,3,4} are more economical than {1,2,3,4} because they require one fewer phage. Deciding whether to use {1,2,3} or {2,3,4} might be done on the basis of the ruggedness of Phage 1 compared to Phage 4. In other words, if it is easier to keep a colony of Phage 1 alive than it is to keep a colony of Phage 4 alive, the set {1,2,3} would be preferable to the set {2,3,4}.

Note that not all sets of three phages will distinguish among the strains of bacteria. For example, if we use the phage set {1,3,4}, then Bacteria 4 and Bacteria 5 have the same kill patterns: 1 0 0. Thus, not all sets having the same number of phages have equal properties.

You might wonder about whether it is possible to find a set of 2 phages that distinguishes among the 5 bacterial strains. One approach to answering this question is to examine all six of the 2-element phage sets. Alternatively, one can note that with $N$ different phages at most $2^N$ different kill patterns are possible. Thus, with 2 phages only 4 different kill patterns (00, 01, 10, 11) are possible, so it is impossible to find a set of two phages that distinguish among 5 strains of bacteria.

Real data is distinguished from the data presented in Table 1 in several important respects. First, the amount of data presented is much greater than the amount presented in Table 1. The number of phage types might reach 100 or more and the number of bacterial strains might approach 1000. Furthermore, the kill patterns tend not to be as neat as those shown in Table 1. Some phages kill almost every strain, while others kill a small number of strains.

To further complicate matters, it is generally not feasible to find enough phages to completely distinguish among bacterial strains. This means that the initial data has some rows equal to one another. Of course, subsets of phages can never distinguish among bacterial strains which cannot be distinguished by the entire phage collection. Similarly, it is possible to have different phage strains that kill exactly the same bacterial strains. These phages are redundant from the point of view of a phage typing set.

The computer is essential for checking the data. It can quickly determine which phages produce identical kill patterns and which bacteria are indistinguishable. It can also sort the data so that kill patterns can be checked for duplicates and found more easily in reference tables. Finally, it can search for smallest phage typing sets and be used in a what-if capacity to provide data on the effectiveness of phage typing sets which lack certain phages. This allows scientists to trade-off coverage against costs of keeping phage cultures going. The next section provides some of the background on the problem of finding optimal phage typing sets.

## 4. *NP*-COMPLETE AND *NP*-HARD PROBLEMS

Over the last several decades computational complexity theory has produced a taxonomy of problems and identified key difficult problems. For the purposes of this paper we will consider the hierarchy of problem difficulty as having three components. A much more complete and rigorous treatment of these topics can be found in Garey and Johnson's book [13].

The first component of the problem difficulty hierarchy consists of problems which can be solved by "efficient" algorithms. Here the word efficient means that the algorithms run in polynomial time. The first component is also known as the class $P$.

The second component of the problem difficulty hierarchy consists of problems for which it can be "efficiently" verified that a proposed solution is correct. In other words, at present there are no efficient algorithms known which will produce solutions for all problems in this class, but efficient algorithms are known which can determine whether a proposed solution is correct. This class is called $NP$ and it includes the class $P$.

The outstanding open problem in computational complexity theory is whether the class $P$ is a proper subset of the class $NP$ or whether the two classes are identical. Most practicing computer scientists believe that $P$ is a proper subset of $NP$ and that there are problems in the class $NP$ for which efficient algorithms do not exist.

The pioneering work of Stephen Cook in the early 1970's established the existence of $NP$-complete problems. These are problems which are in a technical sense the "hardest" problems in the class $NP$. This means that if an efficient algorithm can be found for an $NP$-complete problem, then efficient algorithms can be found for all problems in the class $NP$.

Shortly after Cook published his result, Richard Karp showed that many problems of widespread interest were $NP$-complete. This stimulated a lot of research to find other $NP$-complete problems and made people realize that the reason they were not able to find efficient algorithms for some problems was that the problems were $NP$-complete or even harder. Problems that are at least as hard as $NP$-complete problems are called $NP$-hard problems. In some cases it is not known whether these problems are in the class $NP$, while in others it is known that they are not in the class $NP$.

The third component of the problem difficulty hierarchy we have been describing consists of all the problems that are not in the class $NP$. These problems are all $NP$-hard. The rest of this section is devoted to showing the relevance of this theory to our problem of finding optimal phage typing sets.

The book [13, pp. 187–288] by Garey and Johnson provides an extensive list of $NP$-complete problems organized by subject area. When trying to solve a problem it is worthwhile checking whether the problem or its close relatives are listed in Garey and Johnson's book. Often it is necessary to reinterpret the problem you are trying to solve in terms of the problem being described in the book.

To illustrate this procedure and to establish the result that we are seeking in this paper, we will show that the problem of finding an optimal phage typing set is $NP$-hard. Thus, we cannot expect to find an efficient algorithm for solving this problem. Without an efficient algorithm, the solution of large problems will, most likely, require substantial computer power.

Page 222 of [13] describes the *Minimum Test Set Problem* and gives references that show that this problem is $NP$-complete. The problem is the following. You are given a finite set $S$, a collection $C$ of subsets of $S$ and an integer $K$ less than or equal to the size of $C$. The problem is to determine whether there exists a subcollection $C'$ of $C$ having no more than $K$ elements and having the property that for any two distinct elements $u$ and $v$ in $S$ there is a set $X$ in $C'$ which contains exactly one of $u$ and $v$.

A subset which contains exactly one of $u$ and $v$ is said to *distinguish* between $u$ and $v$. A collection of subsets of $S$ that distinguishes between every pair of elements in $S$ is called a *test set*. Thus, this problem asks whether there exists a test set of size less than or equal to $K$.

The Minimum Test Set Problem can be related to the problem of finding optimal phage typing sets as follows. Let $S$ be the set of all bacterial strains. Represent each phage by the set of strains which it kills, and let the collection of sets that results be the collection $C$. A test set $C'$ can be

thought of as a set of phages that distinguishes between any pair of distinct bacteria, since to be a test set means that for any pair of bacteria we can find a phage which kills one but not the other.

According to the work cited in [13] it is an $NP$-complete problem to determine whether a test set having no more than $K$ elements exists. It is clear that finding a smallest test set $C'$ is at least as hard as answering the question of whether there is a test set of size no more than $K$, because if you are given a smallest test set you can count it and then answer questions about whether test sets of size less than or equal to $K$ exist. It follows that finding a smallest phage typing set is $NP$-hard.

From the results cited in [13] it is seems clear that since the problem of finding optimal phage typing sets is $NP$-hard, we are not likely to find an efficient algorithm for solving it. The fact that no general, efficient algorithm exists does not mean that there is no efficient algorithm for particular instances of the problem. The next section describes some of the methods we use for solving this problem and the results that we have been able to obtain on an IBM 3090. We have been fortunate to find optimal solutions efficiently for the cases that we have studied so far.

## 5. FINDING OPTIMAL SOLUTIONS

The availability of a powerful computer inspired us to write programs for finding the optimal phage typing set and to explore more powerful heuristics than we were able to use on earlier generation computers. Our current program is written in VS Pascal and works as follows.

The first step is to read the data from a file. The file format is quite simple. The first line is a file header that contains information about the file for the benefit of humans, but which is ignored by the computer. The second and fourth lines are also headers that are ignored by the computer. The third line lists the smallest numerical label used by a phage and the largest. These low and high values are used to generate the set of all possible phages. The fifth line lists phages that are to be excluded from the set of all possible phages. This makes it easy to experiment with different phages sets if it is necessary to eliminate some unstable phages. Any number of lines may be used to list the excluded phages since the program reads numbers until it encounters $-1$ (valid phage labels must be positive).

Following the excluded phages the data file contains bacteria labels contained on a line followed by a list of the phages that kill that type of bacteria. As with the excluded phages, this data may run over several lines. The data must end with $-1$.

After reading in the data the program eliminates duplicate patterns and phages. As noted earlier, if the original phage set cannot distinguish between two strains of a bacteria, no subset of that original set can distinguish between them. Thus, eliminating redundant rows and columns speeds up the analysis without eliminating any discriminating power.

The program marks and moves the data in such a way that it can be accessed if necessary, yet it is ignored in calculations where it is not needed. These techniques permit an optimal search that does not need excessive memory space.

After eliminating duplicates, the program constructs a set of differences (we will refer to these sets as *difference sets* for every pair of distinct patterns. For a given pair of distinct patterns, the difference set contains all the phages that belong to one of the patterns but not both. Alternatively, it can be defined as the set containing all the phages that are found in exactly one of the kill patterns. Set theoretically, it is the symmetric difference of the two kill patterns.

Finding the difference sets takes a fair amount of time and lots of memory since if there are $N$ different kill patterns, there will be $(N*(N-1))/2$ difference sets. Our program can be rewritten so it does not construct all the difference sets, but this would increase its running time. With the large address space of the IBM 3090 it was decided to construct the difference sets and reduce the running time.

A set of phages is a typing set if it has a non-empty intersection with every difference set. This observation makes it easy to check whether a set of phages is a typing set and also provides the machinery for solving the problem.

Before searching for phage typing sets, the program makes use of the following two observations. First, if a difference set contains just one phage, that phage must be contained in every phage typing set. To see this, note that a difference set contains exactly one phage if the two kill patterns used to construct the difference set were identical except that one pattern had one phage more than the other. The only way to differentiate between two such patterns (strains of bacteria) is to use the extra phage. Second, if whenever $phage_1$ differentiates between a pair, $phage_2$ will also differentiate between a pair we may safely drop $phage_2$ from the analysis.

The program uses the above two observations until they can no longer be applied to find phages that must be in all phage typing sets. In practice this eliminates many difference sets and allows the computer to find the optimal solution in a reasonable amount of time.

After finding phages that must be in all phage typing sets the program uses a greedy algorithm and an optimal algorithm to find phage typing sets. Greedy algorithms do not always produce optimal solutions but they often produce near optimal solutions, and it was considered worthwhile to find out how successful a greedy algorithm would be in solving this problem. This information might be valuable in the future if we encounter a problem that is too large for finding the optimal solution.

The greedy algorithm works as follows. First, it picks the phage which is found in the largest number of difference sets. Then it removes all the difference sets which contain that phage, and repeats the process by selecting the phage that is found in the largest number of the remaining difference sets. Eventually a phage typing set is found. The phage typing set found by the greedy algorithm need not be optimal, but in the cases we analyzed, it produced either an optimal solution or a solution having 1 extra phage.

The optimal algorithm uses recursion to exhaustively search through possible phage typing sets. It always finds the phages that must be in a phage typing set using the observations mentioned above. It then finds the phage that is contained in the largest number of difference sets. It finds the optimal solution that includes that phage as well as the optimal solution that does not include that phage. It picks the true optimal by picking the best of the two solutions. Great care must be taken to mark data so that different recursive calls get the correct difference sets to consider and so that enough phages are around to always produce a phage typing set. Table 2 shows the results we have obtained using our programs.

Table 2. The results of phage typing set finding program.

| Bacteria Type | Number of Strains | Number of Phages | Opt. Sol. Size | Saved | Greedy Sol. Size | In All Sets | Run Time Secs. |
|---|---|---|---|---|---|---|---|
| *Staphylococcus chromogenes* | 38 | 22 | 10 | 55% | 10 | 0 | 1.58 |
| *Escherichia coli* | 529 | 53 | 37 | 30% | 37 | 37 | 20.02 |
| *Staphylococcus epidermidis* | 63 | 22 | 13 | 41% | 13 | 9 | 0.15 |
| *Staphylococcus hyicus* | 54 | 32 | 13 | 59% | 14 | 5 | 3.16 |
| *Salmonellae* | 334 | 50 | 29 | 42% | 30 | 19 | 7.72 |
| *Staphylococcus aureus* | 74 | 34 | 19 | 44% | 19 | 13 | 0.30 |

Column 1 in Table 2 lists the name of the bacteria being typed. Columns 2 and 3 list the distinct numbers of strains and phages with which we need to work. Column 4 lists the size of the smallest possible typing set (there may be more than one typing set of that size). Column 5 lists the percentage of the original phage typing set that is discarded to reach the optimal solution. This percentage ranges from a low of 30% to a high of 59%, showing that the computer analysis produces substantial savings.

Column 6 lists the size of the solution produced by the greedy algorithm. In 4 of the 6 cases the greedy algorithm found an optimal solution. In the remaining 2 cases, the greedy algorithm found a typing set that was just one phage larger than the solution produced by the optimal algorithm.

Column 7 tells how many phages must belong to all phage typing sets. These phages are found by searching for all difference sets which consist of only one phage. This number ranges from 0 (*Staphylococcus chromogenes*) to every phage in the optimal solution (*Escherichia coli*). If every phage in the optimal solution must be in all solutions, just about any algorithm will find the optimal phage typing set.

Column 8 lists the IBM 3090 running time needed to run both the optimal algorithm and the greedy algorithm. The running time is not a simple function of the *size of the problem*, which may be thought of as the *product of the number of strains by the number of phages.* A key factor in the amount of time needed to find a small phage typing set is the number of phages that must be in all typing sets. The larger this number, the more phages can be removed before the search for the optimal solution begins. This, of course, speeds up the search.

For example, consider the entries for *Staphylococcus epidermidis* and *Staphylococcus hyicus.* Note that the problem sizes (63*22 = 1,386 and 54*32 = 1,728) are fairly close, yet it took the IBM 3090 roughly 20 times more time to find small phage typing sets for *Staphylococcus hyicus* as it did for *Staphylococcus epidermidis.* While some of the difference is due to the difference in the data, a key fact seems to be that 9 of the phages used in the *Staphylococcus epidermidis* data set must be in every phage typing set, while only 5 of the phages used in the *Staphylococcus hyicus* data set must be in every phage typing set.

The results in Table 2 are very encouraging since they show that even for large data sets such as *Escherichia coli* the running time is quite reasonable. The biggest problem for large data sets is to get enough space to store the difference sets.

## REFERENCES


1. M. Gershman, Preliminary report: A system for typing *Salmonella thompson*, *Appl. Microbiol.* **23**, 831–832 (1972).
2. M. Gershman, A phage typing system for *Salmonella newport*, *Can. J. Microbiol.* **20**, 769–771 (1974).
3. M. Gershman, A phage typing system for *Salmonella anatum*, *Avian Dis.* **18**, 565–568 (1974).
4. M. Gershman, A phage typing system for *Salmonella binza*, *Public Health Lab* **34**, 97–99 (1976).
5. M. Gershman, A phage typing system for salmonellae: *S. senftenberg*, *J. of Milk and Food Tech.*, **39**(10) 682–683 (1976).
6. M. Gershman, Phage typing system for *Salmonella enteritidis*, Appl. Environ. Microbiol. **32**, 190–191 (1976).
7. M. Gershman, A phage typing system for salmonellae: *S. heidelberg*, *J. of Food Protection*, **40**(1) 43–44 (1977).
8. M. Gershman, Single phage typing set for differentiating *salmonellae*, J. of Clin. Microbiol., **5**(3) 302–314 (1977).
9. M. Gershman and G. Markowsky, Reduced set of phages for typing *salmonellae*, J. Clin. Microbiol. **17**, 240–244 (1983).
10. M. Gershman, G. Markowsky and J. Hunter, Reduced set of phages for typing *Escherichia coli.*, J. Diary Sci. **67**, 1306–1315 (1984).
11. M. Gershman, J.A. Hunter, R.J. Harmon, R.A. Wilson and G. Markowsky, Phage typing set for differentiating *Staphylococcus epidermis.*, *Can. J. Microbiol.* **34**, 1358–1361 (1988).
12. M.H. Adams, *Bacteriophage*, Interscience Publishers, NY (1959).
13. M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, San Francisco (1979).