Modelling constant weight codes using tabu search

J. A. Bland and D. J. Baylis

The Nottingham Trent University, Department of Mathematics, Statistics and Operational Research, Nottingham, UK

In this paper a recently developed combinatorial optimisation technique known as tabu search is used to investigate lower bounds on the optimal sizes of some constant weight codes. A feature of tabu search is the use of a memory facility that inhibits entrapment of the search by local optima, enabling the algorithm to continue searching for global optima. It is shown how tabu search may be formulated in a context appropriate to the modelling of constant weight codes. Furthermore a comparison with results obtained using simulated annealing shows that, in some instances, tabu search may give better (i.e., higher) lower bounds on code sizes.

Keywords: constant weight code, optimisation, tabu search

1. Introduction

Let $\mathscr{C}$ denote a binary block code, which is a subset of $\mathscr{Z}_2^n$, the set of all binary strings of length n. The elements of $\mathscr{Z}_2^n$ are called words and the elements of $\mathscr{C}$ are known as codewords.

In applications, codewords correspond to messages sent over a communication channel, which, in general, is subject to noise. One effect of channel noise is to corrupt messages by changing some 0's to 1's, and vice versa, during transmission, so that a received word is not necessarily a codeword. If $\mathscr{C}$ is a small subset of $\mathscr{Z}_2^n$ then the language (i.e., set of messages) has large redundancy, a feature that enables errors to be detected and sometimes corrected. Code $\mathscr{C}$ is called a t error-detecting (correcting) code if the receiver can detect (correct) all instances of up to t errors in a received word. A parameter that controls the error-processing capability of $\mathscr{C}$ is its minimum distance, d, where d is the minimum Hamming distance, $d_H(\underline{c}_i, \underline{c}_j)$, taken over all pairs of codewords $\underline{c}_i, \underline{c}_j$ $(i \neq j)$. If a receiver "decodes" each received word by interpreting it as the nearest codeword, then $\mathscr{C}$ will be t error-correcting provided $d \geqslant 2t + 1$.[1,2]

In practical terms, desirable features of $\mathscr{C}$ are large d, for good error-correction, large m (where $|\mathscr{C}| = m$), in order to be able to send a large variety of messages, and small n, to reduce the time taken to transmit messages. However, these design requirements are incompatible, and a common approach to code modelling is to find the largest value of m for specified values of n and d.

This approach is adopted in the case where $\mathscr{C}$ is a constant weight code, i.e., where each codeword contains the same number of 1's, denoted by w. (These codes do have practical uses, for example, as delay-insensitive codes for transmission along parallel unsynchronised channels[3].) Thus for specified values of n, d, and w the problem is to find the largest values of m, which, in standard notation, is denoted by $A(n, d, w)$. This is a combinatorial optimisation problem in which very few exact values of $A(n, d, w)$ are known, and, in general, the gap between the best upper and lower bounds is large.[4]

In this paper the above optimisation problem is tackled using a combinatorial optimisation technique called tabu search. The aims of the paper are to formulate tabu search in a context appropriate to the modelling of constant weight codes and then to use a computer implementation to obtain lower bounds on $A(n, d, w)$ for various values of n, d, and w.

2. Problem formulation

Consider a combinatorial optimisation problem $(\mathscr{C}, e)$ in which $\mathscr{C}$ is a constant weight code comprising m codewords each of length n and weight w. Also, e is an integer valued objective function, e: $\mathscr{C} \rightarrow \mathscr{Z}$, such that $\mathscr{C}$ has an associated "cost," $e(\mathscr{C})$. Here $e(\mathscr{C})$ denotes, for $d_H(\underline{c}_i, \underline{c}_j) < d$, the sum of the squared differences between

Address reprint requests to Dr. Bland at The Nottingham Trent University, Department of Mathematics, Statistics, and Operations Research, Burton Street, Nottingham NG1 4BU, UK.

Received 2 February 1996; accepted 17 February 1996.

Appl. Math. Modelling 1997, 21:667–672, November

655 Avenue of the Americas, New York, NY 10010

0307-904X/97/$17.00
PII S0307-904X(97)00019-X

d and $d_H(\underline{c}_i, \underline{c}_j)$, where d is a specified "target" minimum distance for the code $\mathscr{C}$ and $d_H(\underline{c}_i, \underline{c}_j)$ is the Hamming distance between codewords $\underline{c}_i$ and $\underline{c}_j$ $(i, j = 1, 2, \ldots m, i \neq j)$. Hence, formally, we have

$$e(\mathscr{C}) = \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \left[d - d_H(\underline{c}_i, \underline{c}_j)\right]^2 \qquad (1)$$
$$d_H(\underline{c}_i, \underline{c}_j) < d$$

As the above summation contains only terms for which $d_H(\underline{c}_i, \underline{c}_j) < d$ it follows that $e(\mathscr{C}) = 0$ when $\mathscr{C}$ has minimum distance d. Also, since $e(\mathscr{C}) \geq 0$, the value of $e(\mathscr{C})$ may be considered a measure of the failure of code $\mathscr{C}$ to attain d. (The terms in equation (1) are squared so that an ever increasing "penalty rate" is incurred the further the value of $d_H(\underline{c}_i, \underline{c}_j)$ is below d).

As previously stated, for specified values of n and w, the problem is to determine the largest code with a specified minimum distance d. That is, the largest value of m is required such that, for specified values of n, w, and d, codewords $\underline{c}_i \in \mathscr{C}$, $i = 1, 2, \ldots, m$, minimise equation (1) to the extent that

$$e(\mathscr{C}(m, n, d, w)) = 0 \qquad (2)$$

where $\mathscr{C}(m, n, d, w)$ denotes a code characterized by the four shown parameters.

3. Tabu search

Tabu search is a recent optimisation technique that has been used to solve (pure) combinatorial problems such as graph coloring[5] and quadratic assignment,[6,7] as well as specific (applied) problems in areas such as resource scheduling[8] and electronic circuit layout design.[9,10] The algorithm is devised in a way that aids the progressive movement of the search process toward a global optimum (here minimum) of an objective function. A feature of the algorithm is a memory facility that is used to avoid the entrapment of the search by local optima. Detailed source information concerning tabu search may be found in Refs. 11 and 12.

Tabu search involves a sequence of moves from one configuration to another in an appropriate search space. The following explanation gives a general framework for this procedure; a more detailed explanation that incorporates the specific implementation used is given in the following section.

If $\mathscr{S}$ is the set of feasible solutions to a combinatorial optimisation problem, then move h is a mapping defined on $\mathscr{S}$ such that h: $\mathscr{S} \rightarrow \mathscr{S}$ transforms solution $s_a \in \mathscr{S}$ into solution $s_b = h(s_a) \in \mathscr{S}$. Furthermore, associated with $s_a \in \mathscr{S}$ is the set $\mathscr{H}_{s_a}$, where $\mathscr{H}_{s_a}$ comprises those mappings h that are applied to s_a. A consequence of these moves (or mappings) is the generation of neighborhoods. The neighborhood $\mathscr{N}(s_a)$ of solution $s_a \in \mathscr{S}$ is given by

$$\mathscr{N}(s_a) = \{s_b | s_b = h(s_a),\ h \in \mathscr{H}_{s_a}\} \qquad (3)$$

It is assumed that if s_a and $s_b \in \mathscr{S}$, then

$$s_b \in \mathscr{N}(s_a) \Leftrightarrow s_a \in \mathscr{N}(s_b) \qquad (4)$$

The tabu search procedure is such that if $s_a \in \mathscr{S}$ is the current solution (configuration), then the search moves to the allowed solution in $\mathscr{N}(s_a)$ with the lowest (for minimisation) objective function value, s_{min}, say, with "cost" denoted by $q(s_{min})$. Whether or not a solution is allowed is determined by reference to a one-dimensional array, called the tabu list because it contains forbidden, or tabu, solutions.

Once the search has moved to s_{min} (i.e., s_{min} is accepted) it is included in the initially empty tabu list. The tabu list, $\mathscr{T}$, acts as a memory in the sense that it contains a specified number of recently accepted solutions. In other words the search progresses via a sequence of moves; an individual move being from $s_a \in \mathscr{S}$ to $s_{min} \in \mathscr{S}$ where

$$q(s_{min}) = \min_{s_b \in \mathscr{N}(s_a) - \mathscr{T}} q(s_b) \qquad (5)$$

The number of solutions in the tabu list is called the tabu list length, L, and it may be maintained on the basis of a first-in-first-out queue. The tabu list is the main feature of the tabu search algorithm, since it permits the avoidance of entrapment of the search by local minima; by equation (5), moves from s_a to s_{min} such that $q(s_{min}) > q(s_a)$ are possible.

The procedure of generating a neighborhood, accepting the allowed (i.e., non-tabu) solution with the lowest "cost," and updating the tabu list is the most basic tabu search technique, and it may be repeated until, for example, a maximum number of moves is reached or a specified computation time is exceeded. Throughout the search the current overall lowest "cost" is recorded together with the associated solution. If during the search local minima are encountered, then the use of the tabu list enables their avoidance. An additional merit of the tabu list is the prevention of cycling within a subset of $\mathscr{S}$ containing L, or fewer, solutions.

4. Modelling approach

The approach adopted in this paper to model constant weight codes involves the following implementation of the tabu search method. For specified values of m, n, and w the search space is taken to be the following set of (constant weight) codes,

$$\mathscr{S} = \{\mathscr{C} | \mathscr{C} = \mathscr{C}(m, n, w)\} \qquad (6)$$

A move h is then a mapping from one constant weight code to another; h: $\mathscr{S} \rightarrow \mathscr{S}$ such that a particular code $\mathscr{C}_a \in \mathscr{S}$ becomes $\mathscr{C}_b = h(\mathscr{C}_a) \in \mathscr{S}$. In this paper mapping h will change one codeword only and will be identified with a n-tuple, written as $\underline{h}$, so that the kth codeword $\underline{c}_{ak} \in \mathscr{C}_a$,

$k \in \{1,2,\ldots m\}$ becomes $\underline{c}_{bk} \in \mathscr{C}_b$. That is, $\underline{c}_{bk} = \underline{c}_{ak} \oplus \underline{h}$ and $\underline{c}_{bi} = \underline{c}_{ai}$ for all $i \neq k$.

Consider $\mathscr{C}_a$ to represent the current code after l moves of a search comprising a maximum l_{max} moves, i.e., $l \in \{1,2,\ldots l_{max}\}$. The $l+1$st. move of the search will then take code $\mathscr{C}_a$ to code $\mathscr{C}_b$, in which $\mathscr{C}_b$ differs from $\mathscr{C}_a$ in its $l+1$ st. codeword only. At move $m+1$ the first codeword is again changed, and so on. In other words, in general terms, move l changes the kth codeword where $k \equiv l \bmod m$.

In order to produce the neighborhood of $\mathscr{C}_a$, codeword $\underline{c}_{ak} \in \mathscr{C}_a$ has an associated set, $\mathscr{H}_{\underline{c}_{ak}}$, which comprises those n-tuples $\underline{h}$ that are applied to $\underline{c}_{ak}$. Formally the neighborhood of $\mathscr{C}_a \in \mathscr{S}$, where $\mathscr{C}_a = \{\underline{c}_{ai} | i = 1,2,\ldots m\}$, is given by

$$\begin{aligned}\mathscr{N}(\mathscr{C}_a) = \{\mathscr{C}_b | \mathscr{C}_b = \{\underline{c}_{bi} | i = 1,2,\ldots m\} \\ \underline{c}_{bi} = \underline{c}_{ai},\ i \neq k, \\ \underline{c}_{bk} = \underline{c}_{ak} \oplus \underline{h},\ \underline{h} \in \mathscr{H}_{\underline{c}_{ak}}\}\end{aligned} \tag{7}$$

In implementation terms, with a specified minimum distance d, equation (5) becomes

$$e(\mathscr{C}_{min}) = \min_{\mathscr{C}_b \in \mathscr{N}(\mathscr{C}_a) - \mathscr{T}} e(\mathscr{C}_b) \tag{8}$$

where $\mathscr{C}_{min}$ is the accepted non-tabu code in $\mathscr{N}(\mathscr{C}_a)$ and $\mathscr{T}$ contains (previously accepted) tabu codes.

Tabu search is an iterative improvement algorithm, hence a starting solution is required. In this study the starting code is taken to be $\mathscr{C}_0 = \{\underline{c}_{0i} | i = 1,2,\ldots m\}$, where $\underline{c}_{0i} = c_{0i1}, c_{0i2}, \ldots c_{0in}$, with $c_{01j} = 1$, $j = 1,2,\ldots,w$ and $c_{01j} = 0$, $j = w+1, w+2,\ldots,n$. The other initial codewords, $\underline{c}_{0i}$, $i = 2,3,\ldots,m$, are cyclic permutations of $\underline{c}_{01}$ and hence also have weight w. Also the moves applied to $\underline{c}_{ak} \in \mathscr{C}_a$, which comprise set $\mathscr{H}_{\underline{c}_{ak}}$, are given by

$$\mathscr{H}_{\underline{c}_{ak}} = \{\underline{h} | \underline{c}_{bk} = \underline{c}_{ak} \oplus \underline{h},\ d_H(\underline{c}_{ak}, \underline{c}_{bk}) = 2\} \tag{9}$$

Hence, equation (7) simplifies to

$$\begin{aligned}\mathscr{N}(\mathscr{C}_a) = \{\mathscr{C}_b | d_H(\underline{c}_{ai}, \underline{c}_{bi}) = 0,\ i \neq k, \\ d_H(\underline{c}_{ak}, \underline{c}_{bk}) = 2\}\end{aligned} \tag{10}$$

with $|\mathscr{N}(\mathscr{C}_a)| = w(n-w)$. Note that in order to obtain $\mathscr{N}(\mathscr{C}_a)$, given by equation (10), the elements of $\underline{h}$ are such that they effect a swop of a "0" and a "1" in $\underline{c}_{ak}$ so that $\underline{c}_{bk}$ also has weight w.

For a specified value of d (together with n and w) the largest value of m is sought such that, at some stage during the search, equation (2) is satisfied; the search "path" comprises a sequence of accepted non-tabu codes obtained using equation (8). For a particular search the value of m is specified and remains fixed. If at a particular move during the search $e(\mathscr{C}(m,n,d,w)) = 0$, then the value of m is increased by unity and another search is performed.

5. Computational results

The tabu search algorithm outlined in the previous sections was applied to the modelling of some constant weight codes. The particular codes investigated (characterised by their n, d, and w values) are shown in *Table 1*.

With reference to *Table 1*, A_{TS} denotes the largest value of m found by tabu search such that $e(\mathscr{C}(m,n,d,w)) = 0$. For each code in *Table 1*, tabu search was performed with a search comprising a maximum of l_{max} moves, where $l_{max} = 5{,}000$. Furthermore, to inhibit cycling within the code-space a tabu list length of $L = 50$ was found to be suitable. The particular move number and time taken (in seconds) to achieve $e(\mathscr{C}(m,n,d,w)) = 0$ during the search with $m = A_{TS}$ are denoted by l_{best} and t_{TS}, respectively, and are given in *Table 1*.

In order to obtain A_{TS}, for each of the specified values of n, d, and w in *Table 1*, a search with a suitably low value of $m(m_-$ say) was performed (to ensure $e(\mathscr{C}(m,n,d,w)) = 0$ at some move l where $l \in \{1,2,\ldots l_{max}\}$), and then successively repeated with $m := m+1$ until, throughout the search, $e(\mathscr{C}(m,n,d,w)) \neq 0$, where $m = m_+$, say. The value of A_{TS} was then taken to be $m_+ - 1$.

To commence each successive search, with $m = m_-$, $m_- + 1,\ldots m_+$, an initial code is required; the results shown in *Table 1* were obtained using the starting codewords $\underline{c}_{0i}$, $i = 1,2,\ldots m$, detailed in the previous section.

All computational results were obtained using a program written in Pascal run on a VAX-11/785 computer.

6. Discussion

The values of A_{TS} given in *Table 1* represent lower bounds (obtained by the presented implementation of tabu search) on the optimal size of the associated constant weight code. The 7 codes in *Table 1* were also considered by El Gamal et al[13] who used simulated annealing to obtain lower bounds. The results of El-Gamal et al. are denoted by A_{SA} and are shown in *Table 2*. Also shown in *Table 2* are the current best known lower bounds and the theoretical upper bounds, denoted by A_{best} and A_{upper}, respectively.[4]

For comparison purposes the A_{TS} values of *Table 1* are reproduced in *Table 2*. When corresponding values of A_{TS} and A_{SA} are compared in *Table 2* it is observed that it is possible for tabu search to match and exceed lower bounds obtained by simulated annealing; this is the situation for

Table 1. Values of A_{TS}, l_{best}, and t_{TS} for various codes

Code	n	d	w	A_{TS}	l_{best}	t_{TS}
1	22	10	9	23	1,554	463.2
2	23	10	7	15	2,542	512.9
3	23	10	8	21	164	51.6
4	23	10	9	28	733	279.7
5	23	10	11	37	941	453.0
6	24	10	8	25	1,859	621.3
7	24	10	9	36	2,259	998.1

Table 2. Comparison of various bounds on code size

Code	A_{TS}	A_{SA}	A_{best}	A_{upper}
1	23	23	35	57
2	15	18	21	21
3	21	28	33	50
4	28	24	45	87
5	37	39	63	135
6	25	33	38	68
7	36	24	56	119

code 1 and codes 4 and 7, respectively, although neither method obtains the corresponding A_{best} values for these particular codes.

It should be noted that simulated annealing is, like tabu search, a computational search algorithm with the ability to avoid entrapment by local optima. However, unlike tabu search, simulated annealing is a stochastic procedure that accepts candidate solutions on a probablistic basis.[14,15] It is likely, therefore, that the results of El Gamal et al.[13] are best (i.e., highest) values obtained from a series of numerical experiments for each code; however, the researchers do not reveal the number of experiments undertaken nor the execution times for the reported A_{SA} values. In contrast, because tabu search is a deterministic algorithm, the A_{TS} values given in *Table 1* are repeatable, similarly for the values of l_{best} and t_{TS}.

In order to visualise the optimisation process the convergence curve for code 3, for example, is shown in *Figure 1*. The graph in *Figure 1* shows the current overall lowest value of $e(\mathscr{C})$ at each of the 164 moves taken to achieve $e(\mathscr{C}) = 0$ (where $\mathscr{C}$ denotes the constant weight code that gives the current overall lowest value of $e(\mathscr{C})$). As observed in *Figure 1*, there is rapid initial convergence from $e(\mathscr{C}_0) = 2{,}344$ (where $\mathscr{C}_0$ is the start code) for approximately 45 moves, after which the convergence curve levels off and remains at $e(\mathscr{C}) = 8$ for moves 63 to 162.

For a closer inspection of the optimisation process aspects of the search from move 45 are shown in *Figure 2*. With reference to *Figure 2* the boxes indicate the current overall lowest value of $e(\mathscr{C})$ at each move and clearly show local minima of 24, 20, 12, and 8 starting at moves 48, 51, 61, and 63, respectively. The crosses in the graph of *Figure 2* indicate, at each move, the neighborhood lowest value of $e(\mathscr{C})$ (where $\mathscr{C}$ denotes the non-tabu neighborhood constant weight code with lowest "cost"). As seen in *Figure 2* the neighborhood lowest "costs" are higher (or equal to) the current overall lowest "costs" and represent the search's attempts to climb out of local minima.

For code 3 the codewords that comprise code $\mathscr{C}$ for which $e(\mathscr{C}) = 0$ are presented in *Figure 3*. The actual Hamming distances between all pairs of codewords are given in *Figure 4*, which confirms that the minimum distance is 10.

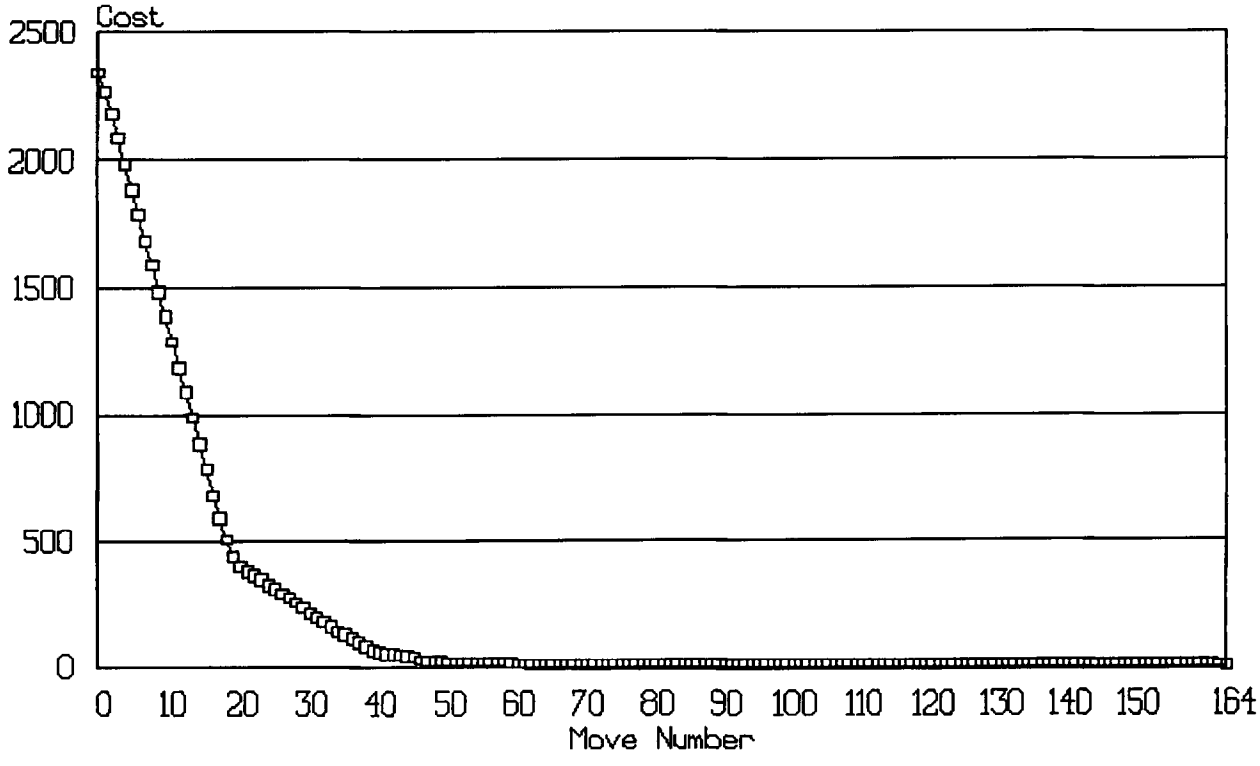

Figure 1. Convergence curve for $\mathscr{C}(21, 23, 10, 8)$.

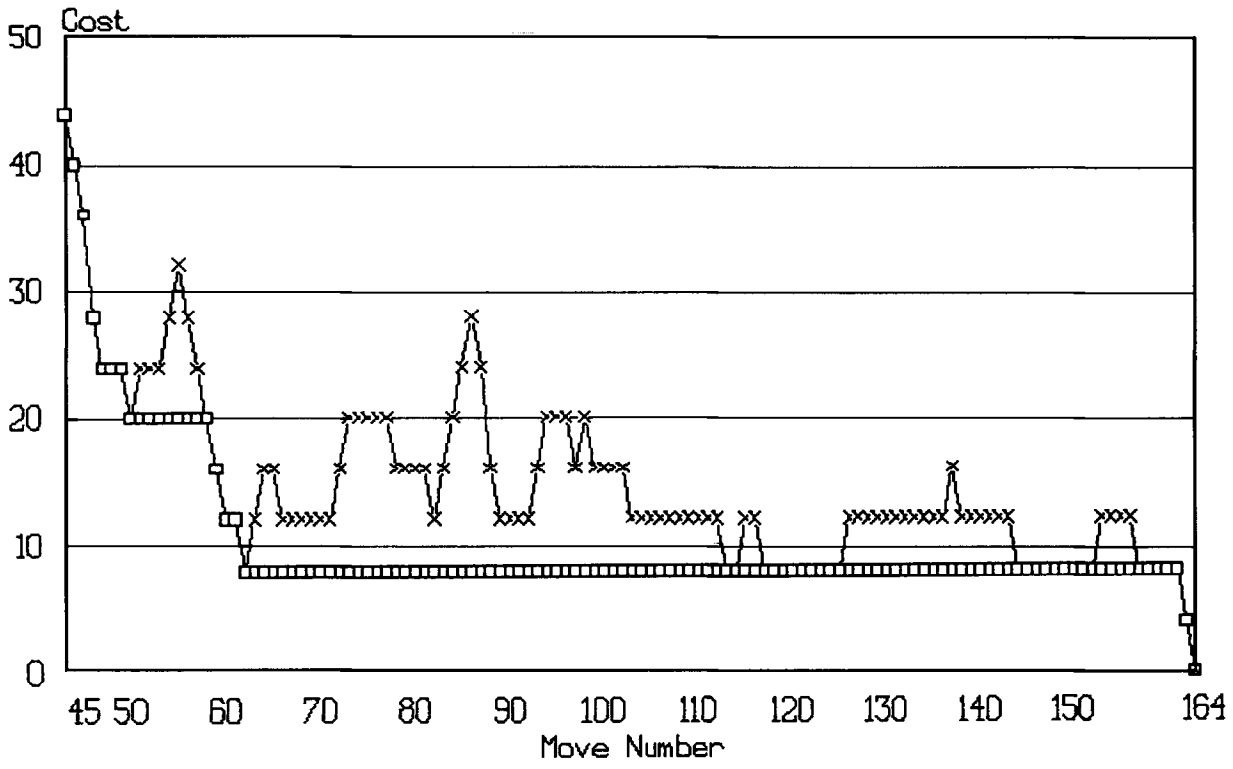

Figure 2. Convergence curve and search history from move 45.

```
10101101000001000010001
11110010000001000100010
10001000111000010010010
00011000001001101011000
10000000011101000100101
00110001001110010001000
01001010000111000001001
00100001101001100000110
01000100010001011001010
10000001110010001001001
00000010101011011100000
11000100000110101100000
01000001000010110010011
00100110010010000011100
11100100000000110001100
00110100101000001111001
01011001010000001100100
01100001010000000111001
10000011000100001010110
00001101100100000101010
00111000010100100000011
```

Figure 3. Code $\mathscr{C}(21, 23, 10, 8)$.

7. Conclusions

Tabu search is a recently developed optimisation technique for discrete variables. In this paper it has been formulated in a context appropriate to the modelling of constant weight codes and then used to investigate lower bounds on the optimal sizes of some particular codes. Computational results show that it is possible, in some instances, for tabu search to match, and also improve on, lower bounds previously obtained by simulated annealing.

As previously explained the strategy employed to obtain the lower bounds (A_{TS}) in *Table 1* made use of the same form of cyclic starting code for successive values of m for each of the 7 codes. However a second starting strategy was investigated. In this case, with $m = m\_$, the cyclic starting code was used, but for subsequent values of m the codewords associated with $e(\mathscr{C}(m-1, n, d, w)) = 0$ were used as the start codewords $\underline{c}_{0i}$, $i = 2, 3, \ldots, m$, for code $\mathscr{C}(m, n, d, w)$. Codeword $\underline{c}_{01}$ was the same as that in the previous strategy. Unfortunately, for the 7 codes investigated, this second strategy did not lead to improved lower bounds, although convergence to the final (nonzero) minimum "cost" was faster. As a final comment on the starting codes (first strategy), for values of m such that $2n \geqslant m > n$, codeword $\underline{c}_{0n+1}$ was used as a generator for subsequent codewords (cyclic permutations), where $\underline{c}_{0n+1} = \underline{c}_{01}$, but with bits $c_{0n+1w} \in \underline{c}_{0n+1}$ and $c_{0n+1w+1} \in \underline{c}_{0n+1}$ interchanged ($\underline{c}_{0n+1} = c_{0n+11}, c_{0n+12}, \ldots c_{0n+1n}$). Values of m such that $m > 2n$ did not occur in this study.

In conclusion it is remarked that the presented tabu search results were obtained from a relatively basic implementation and hence provide encouragement for the further application of the tabu search methodology to combinatorial optimisation problems in coding theory.

0	10	10	10	10	12	10	10	12	10	14	12	10	10	10	10	12	10	10	10	10
10	0	12	12	10	12	10	10	10	14	10	10	12	12	10	12	10	10	10	12	10
10	12	0	10	10	12	14	10	10	10	10	14	10	12	10	12	12	10	10	10	10
10	12	10	0	12	10	10	10	10	12	10	12	12	12	10	12	10	10	12	12	10
10	10	10	12	0	12	10	10	12	10	10	10	14	12	12	12	10	10	10	12	10
12	12	12	10	12	0	10	10	12	10	10	12	10	10	10	10	12	10	12	10	10
10	10	14	10	10	10	0	14	10	10	10	10	10	10	10	14	12	10	12	10	10
10	10	10	10	10	10	14	0	12	12	10	14	10	12	10	10	12	10	10	10	10
12	10	10	10	12	12	10	12	0	10	10	10	10	10	10	10	10	12	12	10	12
10	14	10	12	10	10	10	12	10	0	10	10	10	10	12	10	10	10	10	10	12
14	10	10	10	10	10	10	10	10	10	0	10	12	12	14	10	12	10	12	12	16
12	10	14	12	10	12	10	14	10	10	10	0	10	12	10	12	10	12	10	10	12
10	12	10	12	14	10	10	10	10	10	12	10	0	12	10	12	12	10	10	12	10
10	12	12	12	12	10	10	12	10	10	12	12	12	0	10	10	12	10	10	12	12
10	10	10	10	12	10	10	10	10	12	14	10	10	10	0	10	10	10	12	12	10
10	12	12	12	12	10	14	10	10	10	10	12	12	10	10	0	10	10	12	12	10
12	10	12	10	10	12	12	12	10	10	12	10	12	12	10	10	0	12	10	10	10
10	10	10	10	10	10	10	10	12	10	10	12	10	10	10	10	12	0	14	10	12
10	10	10	12	10	12	12	10	12	10	12	10	10	10	12	12	10	14	0	10	12
10	12	10	12	12	10	10	10	10	10	12	10	12	12	12	12	10	10	10	0	10
10	10	10	10	10	10	10	10	12	12	16	12	10	12	10	10	10	12	12	10	0

Figure 4. Hamming distances.

References

1. Hill, R. *A First Course in Coding Theory*. Oxford University Press, Oxford, 1986
2. Pretzel, O. *Error Correcting Codes and Finite Fields*. Oxford University Press, Oxford, 1992
3. Verhoeff, T. Delay-insensitive codes: An overview. *Distrib. Comput.* 1988, **3**, 1–8
4. Brouwer, A. E., Shearer, J. B., Sloane, N. J. A., and Smith, W. D. A new table of constant weight codes. *IEEE Trans. Inform. Theory* 1990, **IT-36**, 1334–1380
5. Hertz, A. and de Werra, D. Using Tabu Search Techniques for Grah Coloring. *Computing* 1987, **29**, 345–351
6. Skorin-Kapov, J. Tabu search applied to the quadratic assignment problem. *ORSA J. Comp.* 1990, **2**, 33–45
7. Taillard, E. Robust taboo search for the quadratic assignment problem. *Parallel Comput.* 1991, **17**, 443–455
8. Laguna, M., Barnes, J. W., and Glover, F. Tabu search methods for a single machine scheduling problem. *J. Intelligent Manuf.* 1991, **2**, 63–74
9. Bland, J. A. and Dawson, G. P. Tabu search and design optimisation. *Computed-Aided Des.* 1991, 23, 195–201
10. Song, L. and Vanelli, A. A VLSI placement method using tabu search. *Microelectronics*. 1992, **23**, 167–172
11. Glover, F. Tabu search, Part I. *ORSA J. Comp.* 1989, **1**, 190–206
12. Glover, F. Tabu search, Part II. *ORSA J. Comp.* 1990, **2**, 4–32
13. El Gamal, A. A., Hemachandra, L. A., Shperling, I., and Wei, V. K. Using simulated annealing to design good codes. *IEEE Trans. Inform. Theory* 1987, **IT-33**, 116–123
14. Kirkpatrick, S., Gelett, C. D., and Vecchi, M. P. Optimization by simualted annealing. *Science* 1983, **220**, 671–680
15. Cerny, V. Thermodynamical approach to the travelling salesman problem: An efficient simulation algorithm. *J. Optim. Theory Appl.* 1985, **45**, 41–51