Bounds for short covering codes and reactive tabu search

Carlos Mendes a, Emerson L. Monte Carmelo b, Marcus Poggi a,*

a Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Brazil
b Departamento de Matemática, Universidade Estadual de Maringá, Brazil

A R T I C L E   I N F O
Article history:
Received 7 November 2008
Received in revised form 11 May 2009
Accepted 10 November 2009
Available online 16 December 2009

Keywords:
Covering code
Tabu search
Dominating set
Covering radius
Finite field

A B S T R A C T

Given a prime power \( q \), \( c_q(n, R) \) denotes the minimum cardinality of a subset \( H \) in \( \mathbb{F}_q^n \) such that every word in this space differs in at most \( R \) coordinates from a multiple of a vector in \( H \). In this work, two new classes of short coverings are established. As an application, a new optimal record-breaking result on the classical covering code is obtained by using short covering. We also reformulate the numbers \( c_q(n, R) \) in terms of dominating set on graphs. Departing from this reformulation, the reactive tabu search (a variation of tabu search heuristics) is developed to obtain new upper bounds on \( c_q(n, R) \). The algorithm is described and conclusions on the results are drawn; they identify the advantages of using the reactive mechanism for this problem. Tables of lower and upper bounds on \( c_q(n, R) \), \( q = 3, 4, n \leq 7 \), and \( R \leq 3 \), are also presented.

© 2009 Elsevier B.V. All rights reserved.

1. Introduction

Several researchers have conducted significant studies on covering codes: finding applications, using computational tools, studying related problems, or investigating a large numbers of links or results arising from algebra, combinatorics, theory of information (see [6] for an overview).

In this work we focus on the extremal problem induced by short covering, which was recently introduced in [16]. Let \( \mathbb{F}_q \) denote the finite field with \( q \) elements. Given integers \( n \geq 2 \) and \( 0 \leq R \leq n \), define \( c_q(n, R) \) as the minimum cardinality of a subset \( H \) of \( \mathbb{F}_q^n \) such that every word in this space differs in at most \( R \) coordinates from a scalar multiple of a vector in \( H \).

One of the reasons for studying short coverings is their relationships with the classical numbers described as follows. Let \( K_q(n, R) \) denote the minimum cardinality of a set of codewords \( C \) in any (linear or nonlinear) \( q \)-ary code of length \( n \) such that for every word \( x \), there is a codeword \( c \) in \( C \) in which \( x \) and \( c \) disagree in at most \( R \) coordinates. These numbers were posed for \( R = 1 \) by Taussky and Todd [20] from a purely theoretical context but were generalized for arbitrary \( R \) by Carnielli [4].

The so-called football pool problem is an interesting application which became one of the most famous problems in coding theory. In this problem one wants to find the minimum number of bets that will miss at most one game. This corresponds to the case \( q = 3 \) and \( R = 1 \), because \( K_3(n, 1) \) gives us the best way of guaranteeing \( n - 1 \) correct forecasts in a football pool with \( n \) matches. The case where \( n = 6 \) has been object of a tremendous effort in improving the best known lower bound to 71 [14], while its upper bound remains at 73.

Some of the motivations for studying short covering codes are listed below:

- On the basis of theoretical viewpoint, short covering seems to be an algebraic structure richer than classical covering, since short covering is invariant under scalar multiplication. Moreover, short covering has been connected with several algebraic concepts: actions of group, wreath product, and sum-free sets (see [15,16]).
• From a practicable point of view, short coverings provide us with a way to store codes using less memory than the classical ones.

• Results on short covering codes may bring us record-breaking results on classical codes.

Let us now mention the contributions of this work. Two new classes of short coverings are established by using distinct tools and combinatorial constructions. Moreover, a record-breaking result on classical codes is obtained by using a suitable construction from short covering codes, which yields $K_5(10, 7) = 9$. As another goal of this work, a computational approach is developed in order to improve upper bounds on $c_q(n, R)$, by the use of a metaheuristic based on a variation of tabu search, the so-called reactive tabu search. We provide evidence on the quality of this heuristic by presenting a table on upper bounds for $K_q(n, R)$ too.

This paper is organized as follows. In Section 2, both functions $K_q(n, R)$ and $c_q(n, R)$ are described. Two new optimal classes on $c_q(n, R)$ are obtained in Section 3, and the number $K_5(10, 7) = 9$ is determined. We reformulate the short covering problem in terms of dominating set problem in Section 4. Then a description of the designed algorithm is presented in Section 5. Tables with lower and upper bounds on $c_q(n, R)$ for $q = 3, 4, n \leq 7$, and $R \leq 3$ together with a few conclusions are presented in Section 6. Finally, we list the codes obtained in Appendix.

2. Preliminaries

Let us now describe more formally both covering code and short covering code. For $n \geq 2$ and $q \geq 2$, let $V_q^n$ be the set of all words $x = (x_1, x_2, \ldots, x_n)$ (sometimes written in a shorter manner $x_1x_2 \ldots x_n$) with length $n$ and components $x_i$ taken on any alphabet of $q$ symbols. This set becomes a metric space by defining the Hamming distance $d(x, y)$ between the words $x$ and $y$ as the number of components in which $x$ and $y$ differ. The sphere of center $x$ and radius $R$ is denoted by $B(x, R) = \{y \in V_q^n : d(x, y) \leq R\}$. A subset $C$ is an $R$-covering of $V_q^n$ when

$$\bigcup_{c \in C} B(c, R) = V_q^n.$$ 

Thus, the number $K_q(n, R)$ denotes the minimum cardinality of an $R$-covering of $V_q^n$. Main tools and results on these numbers are collected in [6].

On the other hand, a subset $H$ in the vector space $\mathbb{F}_q^n$ is an $R$-short covering of $\mathbb{F}_q^n$ when $\mathbb{F}_q \cdot H = \{\alpha h : \alpha \in \mathbb{F}_q \text{ and } h \in H\}$ is an $R$-covering of $\mathbb{F}_q^n$. In other words, $H$ is an $R$-short covering of $\mathbb{F}_q^n$ if any $x$ in $\mathbb{F}_q^n$ can be written as a sum of a multiple of $h \in H$ and a linear combination of at most $R$ canonical vectors. The induced extremal problem $c_q(n, R)$ is defined as the minimum cardinality of such subset $H$, i.e.,

$$c_q(n, R) = \min |H| : H \text{ is an } R\text{-short covering of } \mathbb{F}_q^n.$$ 

It is worth stating that $H$ is a short covering iff the set that contains $H$ and all their scalar multiples generates a covering of $\mathbb{F}_q^n$. For instance, a minimal 1-short covering of $\mathbb{F}_q^2$ needs just one vector ($c_q(2, 1) = 1$), in contrast to the fact that $q$ vectors are required in a minimal covering of $\mathbb{F}_q^2$ ($K_q(2, 1) = q$).

We recall a few general results on $c_q(n, R)$ given in [16].

**Proposition 1 (Trivial Upper Bound).** For a prime power $q$, $0 \leq R < n$,

$$c_q(n, R) \leq \frac{q^{n-R} - 1}{q - 1}.$$

This bound is sharp at least for $R = 0$.

The lower bound below is similar to the classical sphere covering bound for $K_q(n, R)$. Let $v$ denote the number of vectors in a sphere of $\mathbb{F}_q^n$ with radius $R$, that is, $v = 1 + \sum_{i=1}^{R} \binom{n}{i} (q - 1)^i$.

**Proposition 2 (Sphere covering bound on c).** For a prime power $q$,

$$c_q(n, R) \geq \left\lceil \frac{q^n - v}{(q - 1)v} \right\rceil.$$

The above bound is attained at least in the case where short coverings are derived from Hamming codes, according to Theorem 4. The next result constitutes a systematic way to translate bounds.

**Theorem 3.** For a prime power $q$ and $n > R > 0$,

$$c_q(n, R) + 1 \leq K_q(n, R) \leq (q - 1)c_q(n, R) + 1.$$

Both inequalities are tight at least for $q = 2$, thus $c_2(n, R) = K_2(n, R) - 1$. Theorem 5 below describes another class where the second inequality is sharp.
3. New exact classes

It is not surprising that our knowledge on exact classes is very seldom, since the computation of \( c_q(n, R) \) seems to be as difficult as the classical numbers \( K_q(n, R) \). For the time being, the optimal known classes are reviewed below (see [16]).

**Theorem 4.** For every prime power \( q \), and \( n \) such that \( n = (q^t - 1)/(q - 1) \) for some \( t \),
\[
c_q(n, 1) = \frac{q^{n-1} - 1}{q - 1}.
\]

**Theorem 5.** We have \( c_q(n, n - t) = 1 \) if and only if \( n \geq (t - 1)q + 1 \).

3.1. A combinatorial construction

In order to show a new class, we introduce a few concepts. For \( n \leq q \), consider the subset \( \Gamma \) of \( \mathbb{F}_q^n \) formed by all vectors whose components are pairwise distinct, that is,
\[
\Gamma = \left\{ x = (x_1, x_2, \ldots, x_n) \in \mathbb{F}_q^n : x_i \neq x_j, 1 \leq i < j \leq n \right\}.
\]
Given a vector \( x = (x_1, x_2, \ldots, x_n) \in \Gamma \), the sets \( M = \{x_1, x_2, \ldots, x_{n-2}\} \) and \( N = \{x_{n-1}, x_n\} \) produce a partition of the components of \( n \).

**Theorem 6.** For a prime power \( q \geq 4 \), \( c_q(q, q - 2) = 2 \).

**Proof.** The lower bound comes from Theorem 5. For the upper bound, choose the vectors \( k = (1, \ldots, 1, 1, 1) \) and \( h = (1, \ldots, 1, 1, \sigma) \) of \( \mathbb{F}_q^5 \), where \( \sigma \) denotes any element in \( \mathbb{F}_q \) such that \( \sigma \neq 0 \) and \( \sigma \neq \pm 1 \). We claim that \( H = \{k, h\} \) is a \((q - 2)\)-short covering of \( \mathbb{F}_q^5 \). It is enough to prove that \( \mathbb{F}_q \cdot H \) is a \((q - 2)\)-covering of \( \mathbb{F}_q^5 \). Pick an arbitrary vector \( x \in \mathbb{F}_q^5 \). We analyze a few cases.

Case 1: If there is a letter, say \( a \), which appears at least in two coordinates of \( x \). Thus \( x \) and \( a \cdot k \) disagree at most in \( q - 2 \) coordinates, i.e., \( d(x, a \cdot k) \leq q - 2 \).

Case 2: Otherwise, \( x \in \Gamma \). In this situation, the set \( N = \{x_{n-1}, x_n\} \) has two distinct symbols. Case 2.1: if \( 0 \in N \), say \( x_{n-1} = 0 \). The symbol \( a = x_n/\sigma \) satisfies the properties \( a \in M \) and \( a \in N \), thus \( d(x, ah) = q - 2 \). Case 2.2: if \( 0 \notin M \). If \( \sigma x_{n-1} = x_n \) and \( \sigma a = x_{n-1} \), then \( \sigma = \pm 1 \), which produces a contradiction from the choice of \( \sigma \). We can suppose \( \sigma x_{n-1} \neq x_n \), without loss of generality. The symbol \( a = x_n/\sigma \) is such that \( a \in M \) and \( a \in N \), thus \( d(x, ah) = q - 2 \). The proof is complete. \( \square \)

The above construction does not work out for \( \sigma = \pm 1 \). Indeed, note that \( H = \{(1, 1, 1, 1, 1), (1, 1, 1, 4, 4)\} \) is not a 3-short covering of \( \mathbb{F}_q^5 \) because the vector \((1, 0, 4, 3, 2)\) is not covered by \( \mathbb{F}_5 \cdot H \).

**Remark.** It is worth mentioning that the computation of the corresponding class for classical covering still remains an open problem. The exact values on \( K_q(q, q - 2) \) are recently extended to \( q \leq 10 \), according to Haas et al. [10].

3.2. Short covering codes and covering radius

We begin this section reformulating \( c_q(n, R) \) as a kind of covering radius. Given a subset \( H \) in \( \mathbb{F}_q^n \), the short covering radius \( S(H) \) of \( H \) denotes the smallest integer \( R \) such that \( H \) is an \( R \)-short covering of \( \mathbb{F}_q^n \), that is,
\[
S(H) = \min\{|R| : \text{\( H \) is an \( R \)-short covering of \( \mathbb{F}_q^n \)}\}.
\]
Coversing radius has been one of the most investigated parameters of a code. Vast literature on this topic is reported in Brualdi et al. [3].

This new parameter allows to reformulate \( c_q(n, R) \) in the following way:
\[
c_q(n, R) = \min\{|H| : H \subset \mathbb{F}_q^n \text{ such that } S(H) \leq R\}.
\]
The next result can be derived from properties of short covering radius.

**Proposition 7.** The relationship below holds
\[
c_q(n_1 + n_2, R_1 + R_2 + 1) \geq \min\{c_q(n_1, R_1), c_q(n_2, R_2)\}.
\]

**Proof.** Let \( m = \min\{c_q(n_1, R_1), c_q(n_2, R_2)\} \). An arbitrary vector \( x \) in \( \mathbb{F}_q^{n_1+n_2} \) may be written as \((x^1, x^2) = (\pi^1(x), \pi^2(x))\), where \( \pi^1(x) (\pi^2(x)) \) denotes the projection of \( x \) to first \( n_1 \) coordinates (last \( n_2 \) coordinates). Suppose that \( H \) is an \( R_1 + R_2 + 1 \)-short covering of \( \mathbb{F}_q^{n_1+n_2} \) with \(|H| < m\). The subsets \( H^1 = \pi^1(H) \) and \( H^2 = \pi^2(H) \) satisfy \(|H^i| \leq |H| < m\), \( S(H^i) \geq R_i + 1 \) for \( i = 1 \) or 2. By definition, there is a vector \( x^i \) in \( \mathbb{F}_q^{n_i} \) such that \( d(x^i, \alpha h^i) \geq R_i + 1 \) for any \( \alpha \in \mathbb{F}_q \) and any \( h^i \in H^i \). Note that...
\[ d \left( (x^1, x^2), (\alpha h^1, h^2) \right) = d \left( x^1, \alpha h^1 \right) + d \left( x^2, h^2 \right) \geq R_1 + R_2 + 2, \]

hence \( S(H) \geq R_1 + R_2 + 2 \), which produces a contradiction. \[ \square \]

The motivation of the above result came from the inequality by Bhandari and Durairajan [2]: \( K_q(n_1 + n_2, R_1 + R_2 + 1) \geq \min\{K_q(n_1, R_1), K_q(n_2, R_2)\} \).

As a consequence, another sharp class may be derived from Proposition 7, more precisely.

**Theorem 8.** For every \( n \), \( c_3(3n, 2n-1) = 3 \).

**Proof.** The inequality \( c_3(6, 3) \geq \min\{c_3(3, 1), c_3(3, 1)\} = 3 \) follows from Proposition 7, since \( c_3(3, 1) = 3 \) (see [16]). A recursive application of Proposition 7 yields the desired lower bound. On the other hand, pick the vectors in \( \mathbb{F}_3^3 \):

\[
\begin{align*}
  h_1 &= (1, 1, \ldots, 1, 1), \\
  h_2 &= (2, 2, \ldots, 2, 1), \\
  h_3 &= (1, 1, \ldots, 1, 0).
\end{align*}
\]

The set \( H = \{ h_1, h_2, h_3 \} \) is a \( 2n-1 \)-short covering of \( \mathbb{F}_3^{3n} \). Indeed, given an arbitrary vector \( x \), let \( x^{(a)} \) denote the number of times that the symbol \( a \) appears in \( x \), where \( a = 0, 1, 2 \). Suppose that there is a symbol \( a \) with \( x^{(a)} \geq n + 1 \). It is easy to see that \( d(x, ah_1) \leq 2n - 1 \). Otherwise, \( x^{(a)} = n \) for each \( a \), since \( x^{(0)} + x^{(1)} + x^{(2)} = 3n \). We analyze the symbol in the last coordinate of \( x \): say \( a = x_{3n} \). If \( a \neq 0 \), then \( d(x, ah_2) = 2n - 1 \). If \( a = 0 \), note that \( d(x, h_3) = 2n - 1 \). Therefore, \( H \) is a \( 2n-1 \)-short covering of \( \mathbb{F}_3^{3n} \), which completes the proof. \[ \square \]

The contributions on the corresponding class \( K_3(3n, 2n-1) \) are [2,13].

3.3. An application to classical covering code

**Theorem 9.** We have \( c_5(10, 7) = 2 \).

**Proof.** The lower bound is obtained from Theorem 5. For the upper bound, choose the following vectors in \( \mathbb{F}_5^{10} \):

\[
\begin{align*}
  h_1 &= (1, 1, \ldots, 1, 1, 1, 1) \quad \text{and} \\
  h_2 &= (1, 1, \ldots, 1, 2, 2, 2).
\end{align*}
\]

We claim that \( H = \{ h_1, h_2 \} \) produces a 7-short covering of \( \mathbb{F}_5^{10} \). Pick an arbitrary vector \( x \) in \( \mathbb{F}_5^{10} \). If there is a letter \( a \) in \( \mathbb{F}_5 \) such that \( x^{(a)} \geq 3 \), then \( x \) and \( ah_1 \) disagree at most in 7 coordinates, i.e., \( d(x, ah_1) \leq 7 \). Otherwise, we may suppose that \( x^{(a)} \leq 2 \) for every \( a \) which yields

**Claim 1.** The equality \( x^{(a)} = 2 \) holds for every symbol \( a \) in \( \mathbb{F}_5 \).

Consider the decomposition \( x = (u, v) \), where \( u \in \mathbb{F}_5^7 \) and \( v = \mathbb{F}_3^3 \), and define

\[
A = \{ a \in \mathbb{F}_5 : u^{(a)} = 2 \} \quad \text{and} \quad B = \{ a \in \mathbb{F}_5 : v^{(a)} \geq 1 \}.
\]

A closer look reveals that the sets \( A \) and \( B \) produce a partition of \( \mathbb{F}_5 \), by Claim 1. Since \( 2 \leq |B| \leq 3 \), there are two cases: (1) \( |A| = 2 \) and \( |B| = 3 \), or (2) \( |A| = 3 \) and \( |B| = 2 \).

Case 1: If \( |A| = 2 \), say \( A = \{ a, b \} \). Because \( 2a \neq b \) or \( 2b \neq a \), we may suppose \( 2a \neq b \) without loss of generality. Case 1.1: if \( a = 0 \), take \( c = 2b \). Because \( c \neq 0 \) and \( c \neq b \) we have \( c \in B \), thus \( u^{(b)} = 2 \) and \( v^{(c)} = 1 \), that is, \( d(x, bh_2) = 7 \). Case 1.2: if \( a \neq 0 \), choose the symbol \( c = 2a \), which satisfies \( c \neq a \) and \( c \neq b \), thus \( c \in B \). Because \( u^{(a)} = 2 \) and \( v^{(c)} = 1 \), \( d(x, ah_2) = 7 \) holds.

Case 2: For the case where \( |A| = 3 \), consider \( B = \{ c, d \} \), with \( u^{(c)} = 2 \) and \( v^{(d)} = 1 \). Case 2.1: if \( c = 0 \), take the symbol \( a \) such that \( 2a = d \). Since \( a \neq c = 0 \) and \( d \neq a \), we have \( a \in B \). But \( u^{(a)} = 2 \) and \( v^{(d)} = 1 \), that is, \( d(x, ah_2) = 7 \). Case 2.2: if \( c \neq 0 \), choose the symbol \( a \) such that \( 2a = c \). If \( a \neq d \), then \( u^{(a)} = 1 \) and \( v^{(c)} = 2 \), that is, \( d(x, ah_2) = 7 \). Otherwise, \( a \neq d \), thus \( a \in A \). In this situation, \( u^{(a)} = 2 \) and \( v^{(d)} = 2 \), that is, \( d(x, ah_2) = 6 \).

The proof that \( H = \{ h_1, h_2 \} \) is a 7-short covering of \( \mathbb{F}_5^{10} \) is complete. \[ \square \]

The method is also able to bring us record-breaking bound on classical covering. Indeed, the earlier best result \( K_5(10, 7) \leq 10 \), by Östergård [18], (see table in [12]), may be improved by using short covering construction, as explained in the next result.

**Corollary 10.** We have \( K_5(10, 7) = 9 \).

**Proof.** The lower bound was obtained in [10]. On the other hand, choose the set \( C \) formed by the nine vectors in \( \mathbb{F}_5^{10} \):

| 0000000000 |
| 1111111111 |
| 1111111222 |
| 2222222222 |
| 2222222444 |
| 3333333333 |
| 3333333111 |
| 4444444444 |
| 4444444433 |

Note that \( C = \mathbb{F}_5 \cdot H \), where the set \( H \) is described in Theorem 9. Thus the upper bound follows from Theorems 9 and 3. \[ \square \]
4. Short covering codes and dominating sets in graphs

In this section we recall the dominating set problem in graphs and show how the short code covering problem can be reduced to the former.

Let $G = (V, E)$ be a directed graph with node set $V$ and a collection $E$ of ordered pairs in $V \times V$. As usual, $u$ is adjacent to $v$ (or $v$ is a neighbor from $u$) when $(u, v) \in E$. The node set $U \subseteq V$ is said to be a dominating set of $G$ if, for every node $v$ in $V$, either $v$ is in $U$ or there exists a node $u$ in $U$ such that $u$ is adjacent to $v$ (i.e., $(u, v) \in E$). The dominating set problem is the problem of finding a dominating set in the graph (or digraph) whose size is minimum. This problem is strongly NP-hard (see [7]).

Theorem 11. Short covering codes correspond to a class of the dominating set problem in digraphs.

Proof. A translation of short covering code into graph theory is described below. Given the vector space $\mathbb{F}_q^n$ and $0 \leq R \leq n$, let us construct the digraph $G(n, q, R) = (V, E)$ as follows. For each vector $u$ in $\mathbb{F}_q^n$ we associate a node $u$ in $V$. The edge $e = (u, v)$ is in $E$ if and only there is $\alpha \in \mathbb{F}_q^*$ such that $0 < d(\alpha u, v) \leq R$. The set $E$ is not a symmetric relation for $n \geq 2$. Indeed, $(u, 0) \in E$ for each non-null vector $u$, but $(0, u) \in E$ just when $0 < d(0, u) \leq R$. Note that dominating sets in $G(n, q, R)$ are in one-to-one correspondence with the $R$-short covering codes in $\mathbb{F}_q^n$. Thus, a minimum $R$-short covering code in $\mathbb{F}_q^n$ can be found by solving the minimum dominating set problem in $G(n, q, R)$. □

Remark 12. A minimum $R$-covering code also can be derived from a minimum size dominating set in graph, by making the following modifications in the above construction: (i) take $k$ instead of $q$, (ii) take $V_q^n$ instead of $\mathbb{F}_q^n$, (iii) replace the rule “there is $\alpha \in \mathbb{F}_q^*$ such that $0 < d(\alpha u, v) \leq R$” by the rule “$0 < d(u, v) \leq R$”. (see [5]).

5. Reactive tabu search on short coverings

Carnielli et al. [5], Honkala et al. [11] and Östergård [19] use metaheuristics to find upper bounds for $R$-covering codes. Neither of these works used the reactive variation of TS. This section describes an application of reactive TS for covering codes problems.

Tabu Search (TS), Glover [8,9], is an adaptive procedure that has been applied to a wide range of combinatorial optimization problems. It is a path based search algorithm, since the set of solutions visited by TS forms a path in the solution space. The main goal of TS is to avoid the existence of cycles in this path while inducing a broad search of the solution space as well as an intensive search on promising regions. To move from the current solution to another one, a neighborhood of a solution has to be defined. The moving into this neighborhood is almost always a greedy step. However, we can only move from the current solution to one of the solutions in its neighborhood when the corresponding movement is not tabu, i.e., the movement is not forbidden. We describe below the details of our algorithm to find dominating sets in the graph $G(n, q, R) = (V, E)$.

The classical TS scheme keeps a list (tabu list) of fixed size and allows cycles larger than the list size. Even though this scheme may lead to excellent results, it is still present the possibility to have the search path trapped in a locally minimal region as described in [1]. One idea to overcome this problem, also presented in [1], is a reactive mechanism for TS. It amounts to combine the main elements of a TS with a long term memory of the search that keeps a history of the visited solutions. This memory is used to dynamically control the size of the tabu list augmenting its size when cycling becomes frequent and reducing it when a new region starts to be explored. Besides the dynamic control of the size of the tabu list, the reactive TS uses another mechanism to escape from local minima. It consists in successively applying a random perturbation to the current solution when the number of visited solution becomes over a given threshold. Reactive TS has had a reasonable success in several applications (see [1,17]).

We now present the classical TS scheme and than show how the reaction mechanism is incorporated.

5.1. The basic tabu search scheme

A TS scheme starts by defining its solution space. Here we consider the set of all $2^{|q|}$ subsets of $\mathbb{F}_q^n$ (the vertex set of $G(n, q, R)$). As a consequence, not all elements of this space is a dominating set, i.e. a subset of $V$ is not necessarily a feasible solution to the original problem. This choice implies working on what is, sometimes, named extended solution space. The advantage is to have a natural neighborhood and the disadvantage is to require the use of a penalty mechanism to ensure to frequently hit dominating sets, i.e. feasible solutions.

The natural neighborhood is given by adding a vertex not in the current solution set or removing one vertex from this set. This amounts to having a neighborhood of size $|V| = q^n$, which may be large. The objective function, which is the size of the dominating set, now has a penalty component. This component is the product of a penalty factor and the number of vertices not covered by the current solution. Higher is the penalty factor, smaller are the chances of not hitting a dominating set. Of course, when this penalty factor is too high the search tends to behave as if only dominating set were allowed and the search becomes too restrictive. Therefore, adjusting this penalty factor is an important task in the tuning of this TS since it balances the freedom of the search with the incidence of hitting dominating sets.

Our strategy to deal with this penalty factor (ALPHA) is to start with a small value (MIN_ALPHA), increase in each iteration by a constant (STEP_ALPHA), until it reaches a maximum value (MAX_ALPHA), when its value is reset to the smaller one.
This allows the search to walk freely in the beginning of each such cycle and converge gradually to a feasible solution. After a reasonable amount of experiments, we decide to use the values 0.1, 1.0 and 0.01 for the parameters MIN_ALPHA, MAX_ALPHA e STEP_ALPHA, respectively.

Another main element of this basic TS is the tabu list. Here we store the number of the last iteration where a movement (a vertex switch) was made tabu, for each possible movement which is represented by each vertex in $V$. So, testing whether a movement $m$ is tabu amounts to verify whether the difference between its value in the tabu list and the number of the current iteration is larger than the tabu list size, usually referred as tabu tenure. This tabu tenure value will be the object of the reaction mechanism, which sets its value dynamically, and is described in the next section.

The last main element of a TS is the so-called aspiration criterion, which is set here to the most common one, i.e. a solution can violate the tabu restriction when it improves the value of the best solution found until this point in the search. A pseudo-code description for the basic TS is now presented.

**Procedure of the basic tabu scheme**

**Step 1. Initialization**
- Construct an initial Cover $U$ by adding random selected words, not in $U$, until $U$ is a Cover.
- $\text{BestCode} \leftarrow U$.
- $T[M] \leftarrow 0$, for every possible move $M$.
- $\alpha \leftarrow \text{MIN\_ALPHA}$.
- Initialize Reaction Mechanism.

**Step 2. Main Loop. Repeat Number-of-Iterations times.**
- Update Penalty Factor ($\alpha$).
- Select the best move $M$ that is not tabu or satisfies the aspiration criteria.
- Make a Move $M$ in $U$.
- Set a tabuValue of $M$ in $T$ to $T[M] \leftarrow \text{present-iteration}$.
- If $U$ is a cover and cardinality-of-$U < \text{cardinality-of-BestCode}$.
  - $\text{BestCode} \leftarrow U$.
  - $T$.

**Call Reaction Mechanism.**

The algorithm starts by constructing a random feasible solution $U$ (a dominating set of $G(n, q, R)$). Next, the TS elements are initialized. The main loop is repeated a sufficiently large number of times usually given by the available amount of CPU time. Comparisons regarding time are made with respect to the evolution of the best solution value at each instant of this heuristic execution. This loop adjusts the penalty factor, test for tabu movements and aspiration criteria, updates the best known solution and activates the reaction mechanism, next described.

5.2. The reaction mechanism

The tabu tenure control and an eventual random change in the current solution are the main elements of the reaction mechanism. The main idea is to keep a long term memory keeping track of the solutions visited, the iteration when they were last visited and how many times they became a current solution. The objective is to detect cycling in the search. Each time a solution repeats in the search the interval between visits is calculated. A quick reaction occurs when this interval is smaller than a given threshold, this reaction is to set the tabu tenure to a large value. A slow reaction follows gradually reducing the tabu tenure value, since it is expected that a new search region is attained by the search when it keeps a large tabu tenure value.

Besides this dynamic tabu tenure value control, there is another local minima scape mechanism. This is done by keeping track of the solutions that are visited an excessive number of times. When this number of solutions goes beyond a given threshold, a number of random moves is done on the current solution, with high hopes of going into a region not yet explored.

The combination of these two mechanisms turns the reactive TS an important improvement on the classical TS implementations, improving its results on a number of applications of TS.

We now present a general pseudo-code for the reaction mechanism. To implement it, some data structures and variables are necessary. They are: (i) the set $\text{Visited}$ that keeps information on the solutions visited during the search; (ii) the set $\text{OftenReapeated}$ which stores the solutions excessively visited, and its cardinality is kept in variable $\text{chaotic}$; and (iii) variables $\text{countLastSizeChange}$, that keeps the number of iterations since last tabu tenure change, and $\text{movingAvg}$ which stores the average size of the cycles completed in the search.

The reaction mechanism initialization amounts to just create and initialize these variables and sets. The corresponding algorithm follows.

**Reaction mechanism initialization**
- Create empty sets $\text{Visited}$ and $\text{OftenReapeated}$.
- $\text{chaotic} \leftarrow 0$.
- $\text{countLastSizeChange} \leftarrow 0$. 
- \texttt{movingAvg} \leftarrow 0.
- \texttt{tabuTenure} \leftarrow \texttt{MIN\_TENURE}.

\textbf{Reaction mechanism}
- escape \leftarrow \texttt{true}.
- \texttt{countLastSizeChange} ++.
- \textbf{if} \texttt{U} \textbf{is in Visited}.
  - \texttt{TamCycle} \leftarrow \texttt{current iteration} – \texttt{last visit iteration of U}.
  - Increment the number of visits of \texttt{U} and update the iteration of last visit to \texttt{U}.
  - \textbf{if} \texttt{TamCycle} < \texttt{CYCLE\_MIN}.
    - \texttt{movingAvg} \leftarrow 0.9 \times \texttt{movingAvg} + 0.1 \times \texttt{TamCycle}.
    - \texttt{tabuTenure} \leftarrow \texttt{min} (\texttt{tabuTenure} ∗ \texttt{INCREASE}, \texttt{MAX\_TENURE}).
    - \texttt{countLastSizeChange} \leftarrow 0.
  - \textbf{if} number of visits to \texttt{U} > \texttt{MAX\_VISITS} and \texttt{U} is not in \texttt{OftenRepeted}.
    - Insert \texttt{U} in \texttt{OftenRepeted} and increment \texttt{chaotic}.
  - \textbf{if} \texttt{chaotic} > \texttt{Chaos}.
    - escape \leftarrow \texttt{true}.
- \textbf{else}
  - Insert \texttt{U} in \texttt{Visited} and set the number of visits to \texttt{U} to one and the last iteration of a visit to \texttt{U} to the present iteration.
  - \textbf{if} \texttt{countLastSizeChange} ≥ \texttt{movingAvg}.
    - \texttt{tabuTenure} \leftarrow \texttt{max} (\texttt{tabuTenure} ∗ \texttt{DECREASE}, \texttt{MIN\_TENURE}).
    - \texttt{countLastSizeChange} \leftarrow 0.
  - \textbf{if} escape is true.
    - Clear the \texttt{Visited} set.
    - \texttt{chaotic} \leftarrow 0.
    - Make a random number of moves in \texttt{U}.

The reaction mechanism is invoked after each movement. It first tests whether the current solution has already been visited. When this is true, the interval between visits is computed and if it is smaller than \texttt{CYCLE\_MIN} the quick reaction acts by increasing the tabu tenure value by multiplying it by a constant \texttt{INCREASE} (\texttt{INCREASE} > 1) and updating \texttt{movingAvg} and the set \texttt{OftenRepeted}. When the number of visits to this solution exceeds the constant \texttt{MAX\_VISITS} and the solution is not in \texttt{OftenRepeted} the solution is inserted in the set. A last test then checks whether the cardinality of this set, the value of \texttt{chaotic}, becomes larger than the constant \texttt{Chaos}, when the scope mechanism is triggered to move randomly from the current solution.

When the current solution is a first time visit, the slow reaction mechanism acts by reducing the tabu tenure value in case \texttt{movingAvg} is smaller than \texttt{countLastSizeChange}. This is done by multiplying the tabu tenure value by a constant \texttt{DECREASE} (0 < \texttt{DECREASE} < 1). Besides this, all the operations regarding the just visited solution are executed, so that further visits are detected.

It is clear that implementing this reaction mechanism in a strict way would possibly demand an excessive large amount of memory and/or a very smart data structure regarding compression and access time. Therefore, instead of the real set of solutions, we keep a hash table where the hash function has as parameter a string associated to the solution to be stored. This string is obtained by the concatenation of the number of vertices covered by exactly by \texttt{i} vertices of the current solution for \texttt{i} in the range from 1 to 6. Although this has shown to reasonably differ the solutions in our tests, it certainly generates a number of false positive tests, i.e., several different solutions may be considered as being the same, which triggers the mechanism when it should not trigger.

The implementation also uses constants \texttt{MIN\_TENURE} and \texttt{MAX\_TENURE} to limit the tabu tenure value, therefore avoiding unwanted small and large values.

\section{Computational experience}

The reactive TS algorithm was implemented in C++ programming language and the experiments were executed on an Intel(R) Core(TM)2 Duo with 2.2 GHz. Experiences were held on the classical covering problem and on the short covering problem. The experiences on the classical code covering problem aimed at showing the strength of this reactive tabu search. The objective was to give reasons to believe that the upper bounds that we present for the short covering codes, also dominating sets in graphs, are tight.

The parameters were defined empirically. They were adjusted for each problem and size of instance. The main parameters are \texttt{Max\_Visits}, \texttt{Chaos}, \texttt{Cycle\_Min}, \texttt{Min\_Tenure}, \texttt{Max\_Tenure}, \texttt{Increase} and \texttt{Decrease}. For the first three, values of 10, 5 and 8 were used for both problems. The tabu tenure values ranged from 20 to 400 for the classical problem, while for the short code covering it ranged from 5 to 200. The \texttt{increase} and \texttt{decrease} values were set to 1.2 and 0.9.

These parameters allow the tabu search to have a flexible change of tabu tenure and to escape from unpromising regions with wasting too many iterations. This should lead to obtaining quasi-optimal solutions.

The implemented algorithm starts with an initial code which is generated on the basis of two strategies, namely: (i) random generation: where words are joined randomly to the code until it becomes a feasible solution; (ii) greedy generation: take the set of non-covered words, then a word that covers the maximum number of vertices in this set is added to the code until it becomes a feasible solution. The algorithm runs 10 times for each instance and for each one of these strategies.
Table 1
Bounds on $K_q(n, R)$.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$q$</th>
<th>$R$</th>
<th>NI</th>
<th>TF (s)</th>
<th>TT (s)</th>
<th>RTS</th>
<th>BestKnown</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3</td>
<td>1</td>
<td>5000</td>
<td>0.008</td>
<td>0.148</td>
<td>27</td>
<td>27</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
<td>5000</td>
<td>0.156</td>
<td>0.201</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>1</td>
<td>10000</td>
<td>0.127</td>
<td>0.576</td>
<td>73</td>
<td>73</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>2</td>
<td>10000</td>
<td>0.284</td>
<td>1.260</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>3</td>
<td>10000</td>
<td>0.006</td>
<td>3.380</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>1</td>
<td>100000</td>
<td>4.496</td>
<td>9.400</td>
<td>186</td>
<td>186</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>2</td>
<td>100000</td>
<td>20.141</td>
<td>33.694</td>
<td>34</td>
<td>34</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>3</td>
<td>100000</td>
<td>11.696</td>
<td>94.737</td>
<td>12</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>1</td>
<td>10000</td>
<td>0.168</td>
<td>0.880</td>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>2</td>
<td>10000</td>
<td>2.024</td>
<td>3.068</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>3</td>
<td>10000</td>
<td>0.016</td>
<td>6.788</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>1</td>
<td>100000</td>
<td>2.928</td>
<td>15.365</td>
<td>256</td>
<td>256</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>2</td>
<td>100000</td>
<td>82.371</td>
<td>90.709</td>
<td>17</td>
<td>17</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>3</td>
<td>100000</td>
<td>72.996</td>
<td>147.849</td>
<td>16</td>
<td>14</td>
</tr>
</tbody>
</table>

Table 2
Bounds on $c_q(n, R)$.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$q$</th>
<th>$R$</th>
<th>NI</th>
<th>TF (s)</th>
<th>TT (s)</th>
<th>UB</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>3</td>
<td>1</td>
<td>5000</td>
<td>0.004</td>
<td>0.192</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>1</td>
<td>100000</td>
<td>9.788</td>
<td>10.100</td>
<td>37</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>2</td>
<td>100000</td>
<td>0.036</td>
<td>25.593</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>1</td>
<td>100000</td>
<td>10.104</td>
<td>16.645</td>
<td>93</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>2</td>
<td>100000</td>
<td>54.467</td>
<td>66.148</td>
<td>17</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>3</td>
<td>100000</td>
<td>6.132</td>
<td>172.827</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>1</td>
<td>10000</td>
<td>0.004</td>
<td>0.384</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>1</td>
<td>10000</td>
<td>0.148</td>
<td>1.512</td>
<td>21</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>2</td>
<td>10000</td>
<td>2.328</td>
<td>5.932</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>1</td>
<td>100000</td>
<td>23.077</td>
<td>24.345</td>
<td>85</td>
</tr>
<tr>
<td>6</td>
<td>4</td>
<td>2</td>
<td>100000</td>
<td>39.874</td>
<td>115.419</td>
<td>21</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>1</td>
<td>10000000</td>
<td>4932.12</td>
<td>6117.92</td>
<td>341</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>2</td>
<td>10000000</td>
<td>1073.673</td>
<td>2720.241</td>
<td>63</td>
</tr>
</tbody>
</table>

6.1. Classical covering problem

Finding upper bounds for $R$-coverings is often a hard task. Table 1 presents upper bounds on $K_q(n, R)$. In this table NI represents the number of iterations, TF indicates the running time (in seconds) to find the best solution, TT stands for the total running time. The label RTS indicates the upper bound obtained by the proposed reactive tabu search algorithm, while BestKnown denotes the best updated upper bound for each instance, according to Kéri [12]. The choices of the $R$-covering instances to test followed their relevance on related work.

The importance of presenting this table on the classical covering problem is also to allow a comparison between their cardinalities and the corresponding short covering ones. A question to be answered is how much is the reduction on the sizes of the codes when the short version is considered.

A closer look on the above table reveals that our algorithm was able to reach the best known upper bound for 12 among 14 instances, spending relatively little computational time. Since the impact of our approach on the classical codes seems to be as good as possible, it is expected that a similar performance should also hold for the short covering code problem.

6.2. Short covering problem

Table 2 presents the upper bounds obtained on $c_q(n, R)$ by our reactive tabu search. In this table, UB indicates the best upper bound found.

In order to help us to analyze the results, Tables 3 and 4 present both lower and upper bound on $c_q(n, R)$, for $q = 3, 4$, where $2 \leq n \leq 7$ and $1 \leq R \leq 3$.

Let us discuss our computational results on short covering codes, according to the Tables 3 and 4. Besides improving upper bounds given by Proposition 1 and Theorem 3, near-optimal bounds were reached for several instances, and the gap between lower and upper bounds was almost closed for a few ones. Moreover, there are exact solutions too, whose values meet the lower bounds given in Theorems 4 and 8. These instances bring us evidences about the correctness and quality of the proposed approach.

We conclude this work with a comparative analysis between short and classical coverings. Proposition 1 allows us to see that short covering needs less memory than the classical one. Moreover, the cardinality of short covering may decrease in
Table 3
Boundson $c_3(n, R)$.

<table>
<thead>
<tr>
<th>n</th>
<th>$R = 1$</th>
<th>$R = 2$</th>
<th>$R = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>$3^a$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>$4^b$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>$13^c$</td>
<td>$4^d$</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>$35^e$–$37^f$</td>
<td>$7^g$–$8^f$</td>
<td>$3^c$</td>
</tr>
<tr>
<td>7</td>
<td>$78^h$–$93^i$</td>
<td>$13^j$–$17^i$</td>
<td>$5^k$–$6^i$</td>
</tr>
</tbody>
</table>

The unmarked instances follow from Theorem 5.

a Ref. [16].
b Theorem 4.
c Theorem 8.
d $c_4(n + 1, R + 1) \leq c_3(n, R)$, Ref. [16].
e Lower bound from Theorem 3 by using the tables in Ref. [12].
f Upper bound from Table 2 (see the codes in the Appendix).

Table 4
Boundson $c_4(n, R)$.

<table>
<thead>
<tr>
<th>n</th>
<th>$R = 1$</th>
<th>$R = 2$</th>
<th>$R = 3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>$2^a$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>$8^b$–$10^c$</td>
<td>$2^d$</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>$21^e$</td>
<td>$5^f$</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>$76^g$–$85^h$</td>
<td>$11^i$–$21^c$</td>
<td>$3^j$–$5^c$</td>
</tr>
<tr>
<td>7</td>
<td>$254^k$–$341^l$</td>
<td>$27^m$–$63^e$</td>
<td>$5^n$–$14^e$</td>
</tr>
</tbody>
</table>

The unmarked instances follow from Theorem 5.

a Ref. [16].
b Theorem 4.
c $c_4(n + 1, R + 1) \leq c_3(n, R)$, Ref. [16].
d Lower bound from Theorem 3 by using the tables in Ref. [12].
e Upper bound from Table 2 (see the codes in the Appendix).
f Theorem 6.

contrast to the cardinality of the classical version. Indeed, both numbers and gaps in Tables 3 and 4 are significantly less than those in [12]. It is worth mentioning that the average $c_q(n, R)/K_q(n, R)$ is close to $q - 1$ for several instances, and this phenomena may also be interesting for large values of $q$.

Acknowledgements

The first author was partially supported by CAPES. The second and third authors were partially supported by CNPq. We also thank an anonymous referee for valuable comments that improved this paper.

Appendix

The best short covering codes obtained by our computation search in Table 2 (symbol $f$ in Tables 3 and 4) are listed below. Note that $F_3$ corresponds to $Z_3$. Instead of the standard notation $F_4 = \{0, 1, w, w + 1\}$, where $w^2 = w + 1$, we consider $F_4 = \{0, 1, 2, 3\}$, where the symbols follow the rules of multiplication: $0 \cdot x = 0$ and $1 \cdot x = x$ for any $x$, $2.2 = 3$, $3.3 = 2$, and $2.3 = 3.2 = 1$.

List $q = 3$:

$c_3(5, 1) \leq 13$

\[
00112 \ 01011 \ 01120 \ 01202 \ 10111 \ 12010 \ 12201 \ 20001 \ 20110 \ 21211 \ 22012 \ 22121 \ 22200
\]

$c_3(6, 1) \leq 37$

\[
000122 \ 001002 \ 001110 \ 001002 \ 010202 \ 010220 \ 011012 \ 011021 \ 011121 \ 011221 \ 012011 \ 012101 \ 012200 \ 010102 \ 010210 \ 012111 \ 110011 \ 110211 \ 111021 \ 111111 \ 112222 \ 120022 \ 120211 \ 121011 \ 121111 \ 122000 \ 122111 \ 200021 \ 202002 \ 201212 \ 202102 \ 212211 \ 220210 \ 222120
\]
\( c_3(6, 2) \leq 8 \)

\[
\begin{array}{cccccccccccc}
021022 & 022210 & 100112 & 101201 & 211010 & 212222 & 220101 & 221212 \\
\end{array}
\]

\( c_3(7, 1) \leq 93 \)

\[
\begin{array}{cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
References