1 Problem description:

Let n be a natural number and denote by $\mathbb{B}$ the set $\{0,1\}$. For $u, v \in \mathbb{B}^n$ let $\Delta(u,v)$ be the set of positions in which u and v differ, e.g.,

$$\Delta(001, 010) = \{2, 3\}.$$

Let $s = s_1 s_2 \ldots s_l$ be a finite sequence of vectors of $\mathbb{B}^n$. We say that s is a PI-sequence if the following property holds:

$$\forall 1 \le i < j \le l \,:\, \Delta(s_i, s_{i+1}) \cap \Delta(s_i, s_j) \not\subseteq \Delta(s_j, s_{j+1}).$$

Finally, denote by $L(n)$ the maximal length of any PI-sequence on $\mathbb{B}^n$.

The question we are interested in is if $L(n)$ is bounded from above by the Fibonacci numbers, i.e., if

$$L(n) \le F(n+1) \text{ where } F(0) = 1, F(1) = 1, \text{ and } F(n+2) = F(n+1) + F(n).$$

Example: For $n = 3$, resp. $n = 4$ a longest PI-sequence is

$$\begin{array}{ccccccccc}
0 & -- & 1 & -- & 0 & -- & 1 & & 1 \\
0 & -- & 1 & -- & 0 & & 0 & & 0 \\
0 & -- & 1 & & 1 & & 1 & -- & 0
\end{array}$$

resp.

$$\begin{array}{ccccccccccccccc}
0 & -- & 1 & -- & 0 & -- & 1 & -- & 0 & -- & 1 & & 1 & & 1 \\
0 & -- & 1 & -- & 0 & -- & 1 & & 1 & & 1 & -- & 0 & & 0 \\
0 & -- & 1 & -- & 0 & & 0 & & 0 & & 0 & & 0 & -- & 1 \\
0 & -- & 1 & & 1 & & 1 & -- & 1 & & 0 & & 0 & & 0
\end{array}$$

Remark: We may consider any PI-sequence as a matrix. Then permutating the rows of a PI-sequence yields obviously a PI-sequence again. Further, as we are intereseted only in the difference vectors $\Delta(s_i, s_j)$ of two columns, flipping all bits of a row preserves the PI-sequence property. From the definition of PI-sequence it also follows that if a PI-sequence $s_1 s_2 \ldots s_l$ does not start by flipping all bits, then $\bar{s}_1 s_1 s_2 \ldots s_l$ is a also a PI-sequence (with $\bar{s}_1$ the vector we obtain from s_1 by flipping all bits.) We therefore can always assume that a longest PI-sequence starts with the vector consisting only of zeros and that in the first step of a longest PI-sequence are bits are being flipped.

2 Why is this interesting?

Intuition: The behaviour of systems that react to their enviroment, e.g., the inputs by some user, can often be described by means of a game (like parity games, mean payoff games, Markov decision processes). In order to (roughly) describe the underlying intuition, imagine we are given a software system and two classes of persons interacting with it: the users and the system operators. While the users behave quite selfish and do not really care about stability of the system as long as they get their job done, the system operators have to ensure that the system stays stable and operable. Depending on the current state of the system either the users or the system operators can issue a command to the system. The system then reacts to this command and changes its state accordingly. The question then is if for a given software system and some given specification, the system operators have a *strategy* to control the system in such a way that no matter what the users do the system behaves as specified. In many cases it can be shown that it suffices that this strategy is a simple function which tells the system operators for every state controlled by them a single command to be issued always in the state.

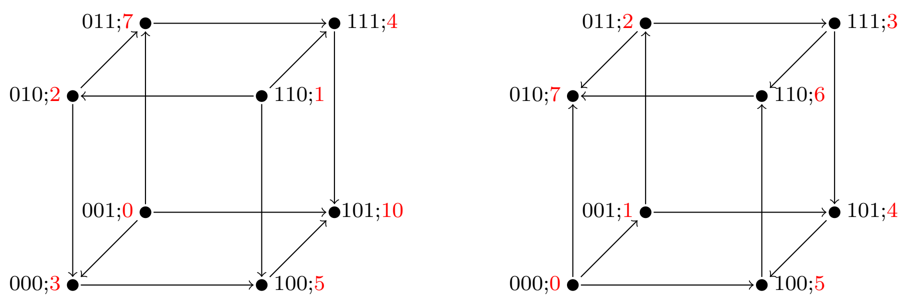
For simplicity, assume in every state where the system operators can issue a command, they can only choose from exactly two possible action (we can always enforce such a normal form). Further assume that the system has only finitely many states (although a restriction, many interesting problems still have this property) and that n is the number of states controlled by the system operators. Then any element of $\mathbb{B}^n$ represents such a strategy. The goal is then to find the best strategy $x^* \in \mathbb{B}^n$. For this, one extracts from the system description an injective *ranking* (function) $r : \mathbb{B}^n \to \mathbb{N}$ wich ranks the strategies s.t. x^* is the strategy with the maximal rank. In general, calculating r for a given strategy $x \in \mathbb{B}^n$ takes time polynomial in the description of the system. Still, a brute force approach for finding x^* would mean a exponential time algorithm in the worst case. Therefore, many heuristics have been proposed for iteratively improving a given strategy until the optimal strategy is obtained. One of the most commonly proposed heuristic is sometimes called "all profitable switches algorithm" (APSA):

Given a strategy $x \in \mathbb{B}^n$, for $i = 1, 2, \ldots, n$, change x at exactly position i. If the resulting strategy has higher rank than x, include i in the set I. The next strategy is then obtain by inverting x at exactly the position included in I.

Abstraction: It is an open problem to obtain a sharp upper bound on the number of iterations done by the APSA. In order to study this question, one usually forgets about the original game and only focuses on the problem of determining the argument x^* maximizing a given injective ranking $r : \mathbb{B}^n \to \mathbb{N}$. It is *conjectured* that every sequence of strategies obtained by the APSA is a PI-sequence if the ranking function is *completely unimodal.* (For now, note that the rankings obtained from the mentioned games like parity games, mean payoff games or Markov decision processes have this property. A definition of "completely unimodal" is given below.)

It therefore follows that the function $L(n)$ (defined in the previous section) is an upper bound on the number of iterations done by the APSA when considering strategies of dimension n. Currently, it is only known that the APSA takes at most $O(\frac{2^n}{n})$ iterations for n-dimensional strategies. So, showing that $L(n) \in O(F(n))$ would be an improvement as $F(n) \in O(1.62^n)$.

Visualization via hypercubes: We can visualize a ranking $r : \mathbb{B}^n \to \mathbb{N}$ by means of an oriented n-dimensional hypercube. The strategies $\mathbb{B}^n$ become the nodes of the hypercube and edges are assumed to point from the strategy/node of lesser rank to the one of greater rank. For an example, consider the following picture where nodes are labeled by both the associated strategy (as binary words of length 3) and its rank (in red):



We then say that an injective ranking r is *completely unimodal* if for *any* face, i.e., sub-hypercube, of the hypercube there is a unique sink. (A sink is a node which only has incoming edges.)

In above examples, only the ranking function depicted on the right is completely unimodal; the one depicted on the left has two sinks for the face defined by the nodes $000, 001, 011, 010$.

For the right ranking function, we obtain the following APSA-sequence when starting in 000:

$$000, 111, 100, 110, 010.$$

3 Goals of the Bachelor thesis:

Up to now, no upper bound on the length PI-sequences is known except for the $O(\frac{2^n}{n})$-bound. Previous experiments using computers to enumerate all possible PI-sequences for a fixed n have only be tractable for $n \leq 7$. For $n = 7$ enumerating all PI-sequences took more than three months using a parallel implementation running on 12 cpus.

The main goal of this Bachelor thesis is to valuate, improve and extend the current parallelization used in the software tool for enumerating the PI-sequences by allowing to only choose some subset of PI-sequences at random. This new tool should then be used to gain some experimental data for $n > 7$. The Bachelor thesis should give also a thorough survey on the theoretical background of the considered problem.

4 References:

- The original source for PI-sequences introduced by Hunter.
- A comparison of different heuristics for maximizing completely unimodal functions by Björklund, Sandberg, Vorobyov.
- Deduction of the $O(2^n/n)$-upper bound by Mansour, Singh.
- The definition of the Fibonacci Seesaw heuristic mentioned in the slides by Hunter.