Hardness of Mastermind

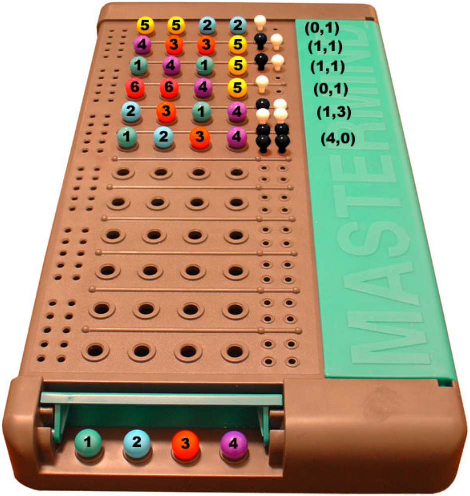
Giovanni Viglietta

Department of Computer Science, University of Pisa, Italy

Venice - June 5$^{\text{th}}$, 2012

"Easy to learn. Easy to play. But not so easy to win."

Mastermind commercial, 1981

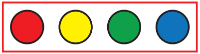


Mastermind is played on a board with colored pegs.
A *codemaker* chooses a secret sequence of colors,
and a *codebreaker* has to guess it in several attempts.

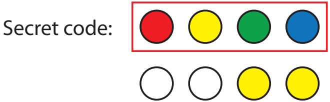
- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.

- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.

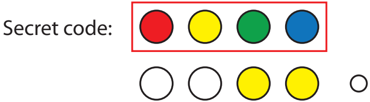
Secret code:



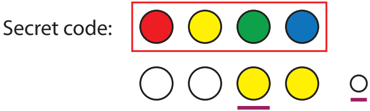
- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.



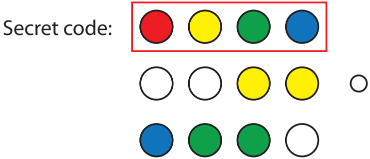
- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.



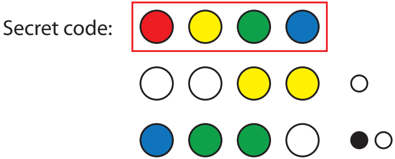
- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.



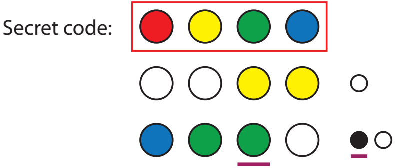
- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.



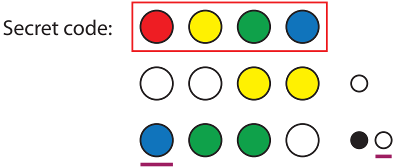
- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.



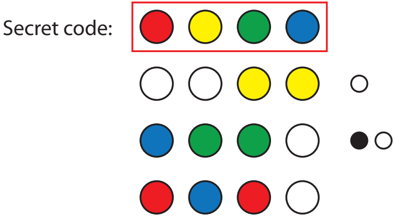
- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.



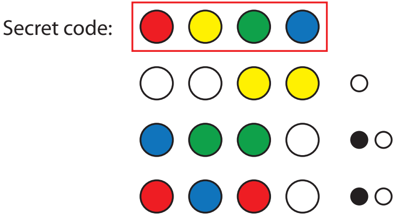
- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.



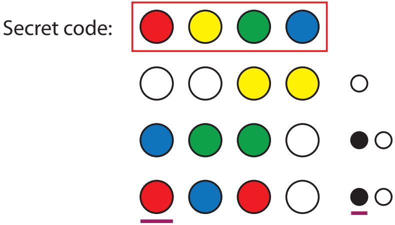
- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.



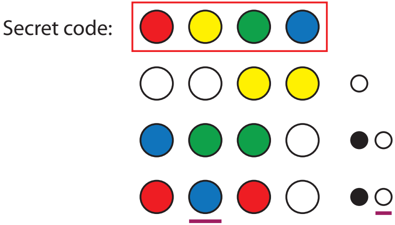
- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.



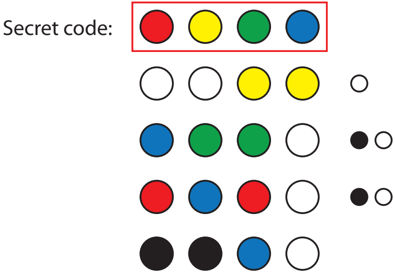
- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.



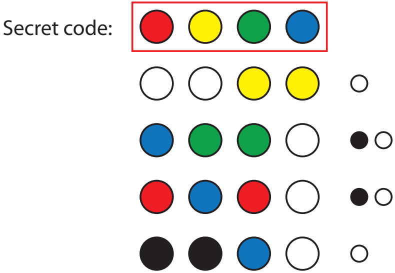
- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.



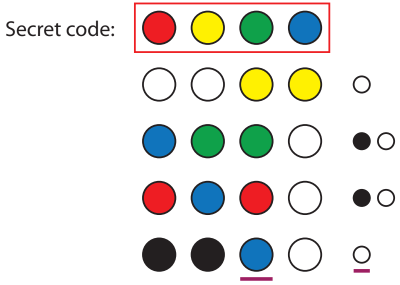
- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.



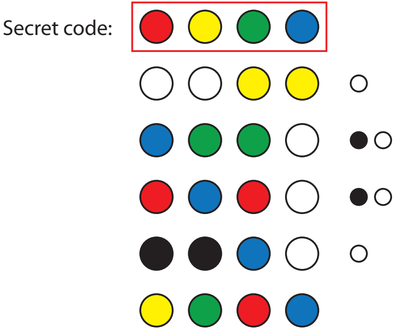
- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.



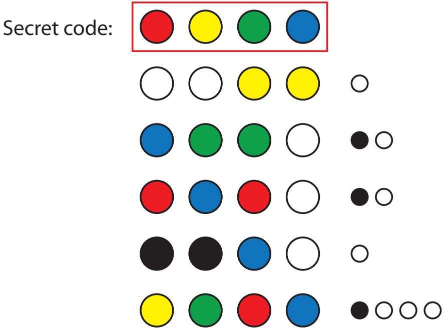
- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.



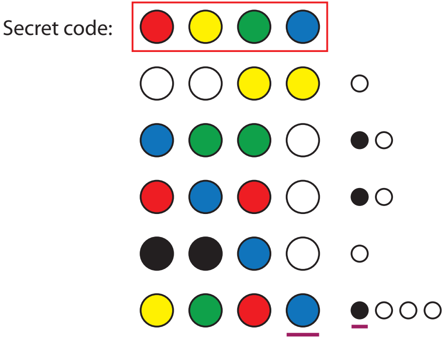
- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.



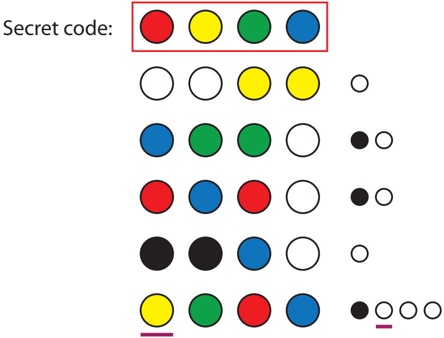
- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.



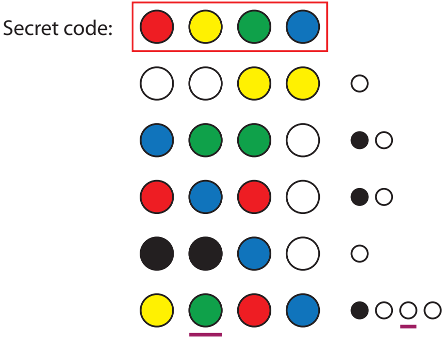
- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.



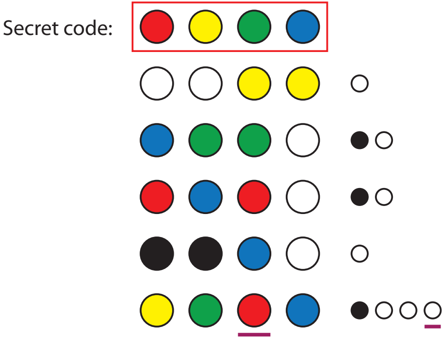
- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.



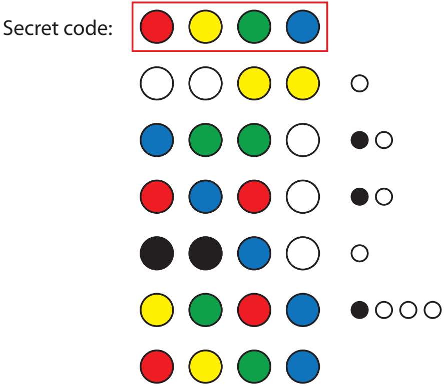
- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.



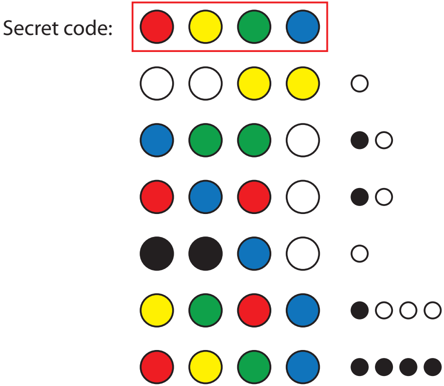
- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.



- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.

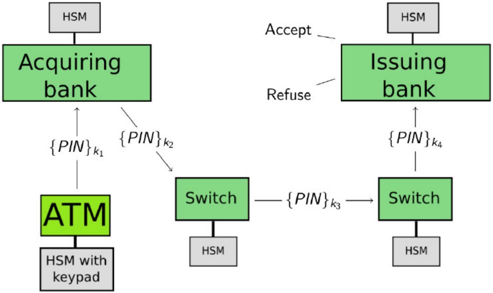


- After each guess, the codemaker responds with some black and white pegs.
  - Black pegs represent correct pegs in the codebreaker's guess that are also well-placed.
  - White pegs represent pegs in the codebreaker's guess that are correct but misplaced.
  - Black and white pegs do not mark the positions of the correct pegs in the codebreaker's guess, but only their amount.



Mastermind in bank frauds

- The relevance of Mastermind in real-life security issues was pointed out in 2010 by *Focardi* and *Luccio*.
- An insider of a bank who gains access to some switch is able to issue several PIN verification API calls, eventually deducing user PINs, digit by digit.
- This kind of attack is performed exactly as an extended Mastermind game played between the insider and the bank's computers.



A feasible heuristic

- A systematic study of Mastermind was carried out by *Chvátal*, in a 1983 paper dedicated to Erdős on his 70^{th} birthday.

- Chvátal suggested a simple divide-and-conquer strategy for the codebreaker to guess the code in $2n \lceil \log c \rceil + 4n + \lceil \frac{c}{n} \rceil$ attempts. Each guess can be computed in polynomial time.

- This bound was subsequently lowered by a constant factor, with an improvement on the same basic idea.

Mastermind: a piece of cake?

- Does Chvátal's strategy trivialize the game?

Mastermind: a piece of cake?

- Does Chvátal's strategy trivialize the game?
- Not really, as long as the number of attempts is critical.
- The classic $(4, 6)$-Mastermind is solvable within 5 guesses, while Chvátal's algorithm guesses 18 times.
- Playing perfectly is still hard.

Exhaustive searches

- Another thread of heuristics was started in 1976 by *Knuth*, who devised a worst-case optimal (w.r.t. the number of guesses) greedy strategy to beat $(4,6)$-Mastermind.

- Every step of the strategy is a brute-force search among all possible guesses and all possible responses of the codemaker.
- The heuristic is based on choosing the guess that will minimize the number of eligible solutions, in the worst case.
- This is practical and optimal for $(4,6)$-Mastermind, but still infeasible and suboptimal in general.
- Several other approaches were adopted, most notably genetic algorithms, achieving different performance tradeoffs.

Satisfiability

Mastermind Satisfiability Problem (MSP)

Input: (n, c, Q), *where* Q *is any sequence of guesses and responses in* (n, c)*-Mastermind.*
Output: YES *if there exists a code that is compatible with all the queries in* Q, NO *otherwise.*

Satisfiability

Mastermind Satisfiability Problem (MSP)

Input: (n, c, Q), *where* Q *is any sequence of guesses and responses in* (n, c)*-Mastermind.*
Output: YES *if there exists a code that is compatible with all the queries in* Q, NO *otherwise.*

- In 2005 *Stuckman* and *Zhang* proved that MSP is NP-complete.
- In 2009 *Goodrich* proved the same result for a variant of Mastermind where the codemaker only responds with black pegs (with an application to genetics).

Solution uniqueness and Restricted Mastermind

Problem (Stuckman–Zhang, 2005)

Can we detect MSP instances with a unique solution?

Solution uniqueness and Restricted Mastermind

Problem (Stuckman–Zhang, 2005)

Can we detect MSP instances with a unique solution?

- Inspired by Goodrich, we define two variants of MSP.
  - In *MSP-BLACK* the codemaker responds only with black pegs.
  - In *MSP-WHITE* the codemaker responds with $(b + w)$ white pegs whenever in MSP he would have responded (b, w). The codebreaker has to guess the code up to reordering of the pegs.

Solution uniqueness and Restricted Mastermind

Problem (Stuckman–Zhang, 2005)

Can we detect MSP instances with a unique solution?

- Inspired by Goodrich, we define two variants of MSP.
  - In *MSP-BLACK* the codemaker responds only with black pegs.
  - In *MSP-WHITE* the codemaker responds with $(b + w)$ white pegs whenever in MSP he would have responded (b, w). The codebreaker has to guess the code up to reordering of the pegs.
- *c-MSP* is always played with c colors, while n is still a variable.
  - Similarly for *c-MSP-BLACK* and *c-MSP-WHITE*.

Solution uniqueness and Restricted Mastermind

Problem (Stuckman–Zhang, 2005)

Can we detect MSP instances with a unique solution?

- Inspired by Goodrich, we define two variants of MSP.
  - In *MSP-BLACK* the codemaker responds only with black pegs.
  - In *MSP-WHITE* the codemaker responds with $(b + w)$ white pegs whenever in MSP he would have responded (b, w). The codebreaker has to guess the code up to reordering of the pegs.
- *c-MSP* is always played with c colors, while n is still a variable.
  - Similarly for *c-MSP-BLACK* and *c-MSP-WHITE*.
- We will determine which of these restrictions are #P-complete.

Preliminary results

Observation

In (n, c)-Mastermind restricted to white pegs, the codebreaker can guess the code after $c - 1$ attempts.

- He tries all possible colors... Although this is suboptimal.

Observation

$\#c$-*MSP-WHITE* $\in$ *FP*.

- There are only $\binom{n+c-1}{c-1} = \Theta(n^{c-1})$ possible codes to check.

Observation

$\#(c-1)$-*MSP* $\leqslant_{pars}$ $\#c$-*MSP*.

- Add the guess $ccc\cdots$ with response $(0, 0)$.
- This holds also for $\#c$-MSP-BLACK and $\#c$-MSP-WHITE.

Hardness of Mastermind

Theorem

$$\#\textit{1-IN-3-SAT} \leqslant_{pars} \begin{cases} \#\textit{MSP-WHITE} \\ \#\textit{2-MSP-BLACK} \\ \#\textit{2-MSP}. \end{cases}$$

- Hence all these variations are #P-complete.

$\#$1-IN-3-SAT $\leqslant_{pars}$ $\#$MSP-WHITE

- Let $\varphi = (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee w) \wedge (y \vee \neg z \vee \neg w)$.
- Colors: $x, \bar{x}, y, \bar{y}, z, \bar{z}, w, \bar{w}, \star$.
- Code length: 4.

$\#$1-IN-3-SAT $\leqslant_{pars}$ $\#$MSP-WHITE

- Let $\varphi = (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee w) \wedge (y \vee \neg z \vee \neg w)$.
- Colors: $x, \bar{x}, y, \bar{y}, z, \bar{z}, w, \bar{w}, \star$.
- Code length: 4.
- Queries:
  - $\star\star\star\star$ with score (0)

#1-IN-3-SAT $\leqslant_{pars}$ #MSP-WHITE

- Let $\varphi = (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee w) \wedge (y \vee \neg z \vee \neg w)$.
- Colors: $x, \bar{x}, y, \bar{y}, z, \bar{z}, w, \bar{w}, \star$.
- Code length: 4.
- Queries:
  - $\star\star\star\star$ with score (0)
  - $x\,x\,\bar{x}\,\bar{x}$ with score (1)
  - $y\,y\,\bar{y}\,\bar{y}$ with score (1)
  - $z\,z\,\bar{z}\,\bar{z}$ with score (1)
  - $w w \bar{w} \bar{w}$ with score (1)

$\#$1-IN-3-SAT $\leqslant_{pars}$ $\#$MSP-WHITE

- Let $\varphi = (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee w) \wedge (y \vee \neg z \vee \neg w)$.
- Colors: $x, \bar{x}, y, \bar{y}, z, \bar{z}, w, \bar{w}, \star$.
- Code length: 4.
- Queries:
    - $\star \star \star \star$ with score (0)
    - $x\ x\ \bar{x}\ \bar{x}$ with score (1)
    - $y\ y\ \bar{y}\ \bar{y}$ with score (1)
    - $z\ z\ \bar{z}\ \bar{z}$ with score (1)
    - $w w \bar{w} \bar{w}$ with score (1)
    - $x\ \bar{y}\ z\ \star$ with score (1)
    - $\bar{x}\ y\ w\ \star$ with score (1)
    - $y\ \bar{z}\ \bar{w}\ \star$ with score (1)

$\#$1-IN-3-SAT $\leqslant_{pars}$ $\#$2-MSP-BLACK

- Let $\varphi = (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee w) \wedge (y \vee \neg z \vee \neg w)$.
- Colors: $\bullet$, $\circ$.
- Code length: 8.

$\#$1-IN-3-SAT $\leqslant_{pars}$ $\#$2-MSP-BLACK

- Let $\varphi = (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee w) \wedge (y \vee \neg z \vee \neg w)$.
- Colors: $\bullet$, $\circ$.
- Code length: 8.
- Queries:

x	$\bar{x}$	y	$\bar{y}$	z	$\bar{z}$	w	$\bar{w}$	Score
$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	(4)

- Let $\varphi = (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee w) \wedge (y \vee \neg z \vee \neg w)$.
- Colors: $\bullet$, $\circ$.
- Code length: 8.
- Queries:

x	$\bar{x}$	y	$\bar{y}$	z	$\bar{z}$	w	$\bar{w}$	Score
$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	(4)

x	$\bar{x}$	y	$\bar{y}$	z	$\bar{z}$	w	$\bar{w}$	Score
$\bullet$	$\bullet$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	(4)
$\circ$	$\circ$	$\bullet$	$\bullet$	$\circ$	$\circ$	$\circ$	$\circ$	(4)
$\circ$	$\circ$	$\circ$	$\circ$	$\bullet$	$\bullet$	$\circ$	$\circ$	(4)
$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\bullet$	$\bullet$	(4)

$\#$1-IN-3-SAT $\leqslant_{pars}$ $\#$2-MSP-BLACK

- Let $\varphi = (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee w) \wedge (y \vee \neg z \vee \neg w)$.
- Colors: $\bullet$, $\circ$.
- Code length: 8.
- Queries:

x	$\bar{x}$	y	$\bar{y}$	z	$\bar{z}$	w	$\bar{w}$	Score
$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	(4)

x	$\bar{x}$	y	$\bar{y}$	z	$\bar{z}$	w	$\bar{w}$	Score
$\bullet$	$\bullet$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	(4)
$\circ$	$\circ$	$\bullet$	$\bullet$	$\circ$	$\circ$	$\circ$	$\circ$	(4)
$\circ$	$\circ$	$\circ$	$\circ$	$\bullet$	$\bullet$	$\circ$	$\circ$	(4)
$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\bullet$	$\bullet$	(4)

x	$\bar{x}$	y	$\bar{y}$	z	$\bar{z}$	w	$\bar{w}$	Score
$\bullet$	$\circ$	$\circ$	$\bullet$	$\bullet$	$\circ$	$\circ$	$\circ$	(3)
$\circ$	$\bullet$	$\bullet$	$\circ$	$\circ$	$\circ$	$\bullet$	$\circ$	(3)
$\circ$	$\circ$	$\bullet$	$\circ$	$\circ$	$\bullet$	$\circ$	$\bullet$	(3)

#1-IN-3-SAT $\leqslant_{pars}$ #2-MSP

- Let $\varphi = (x \vee \neg y \vee z) \wedge (\neg x \vee y \vee w) \wedge (y \vee \neg z \vee \neg w)$.
- Colors: $\bullet$, $\circ$.
- Code length: 8.
- Queries:

x	$\bar{x}$	y	$\bar{y}$	z	$\bar{z}$	w	$\bar{w}$	Score
$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$(4,0)$

x	$\bar{x}$	y	$\bar{y}$	z	$\bar{z}$	w	$\bar{w}$	Score
$\bullet$	$\bullet$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$(4,2)$
$\circ$	$\circ$	$\bullet$	$\bullet$	$\circ$	$\circ$	$\circ$	$\circ$	$(4,2)$
$\circ$	$\circ$	$\circ$	$\circ$	$\bullet$	$\bullet$	$\circ$	$\circ$	$(4,2)$
$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\circ$	$\bullet$	$\bullet$	$(4,2)$

x	$\bar{x}$	y	$\bar{y}$	z	$\bar{z}$	w	$\bar{w}$	Score
$\bullet$	$\circ$	$\circ$	$\bullet$	$\bullet$	$\circ$	$\circ$	$\circ$	$(3,4)$
$\circ$	$\bullet$	$\bullet$	$\circ$	$\circ$	$\circ$	$\bullet$	$\circ$	$(3,4)$
$\circ$	$\circ$	$\bullet$	$\circ$	$\circ$	$\bullet$	$\circ$	$\bullet$	$(3,4)$

Solutions

$$(x \lor \neg y \lor z) \land (\neg x \lor y \lor w) \land (y \lor \neg z \lor \neg w)$$

x	y	z	w
T	T	T	T
T	T	T	F
T	T	F	T
T	T	F	F
T	F	F	T
F	T	T	T
F	T	T	F
F	F	T	F
F	F	F	T
F	F	F	F

x	$\overline{x}$	y	$\overline{y}$	z	$\overline{z}$	w	$\overline{w}$
●	○	●	○	●	○	●	○
●	○	●	○	●	○	○	●
●	○	●	○	○	●	●	○
●	○	●	○	○	●	○	●
●	○	○	●	○	●	●	○
○	●	●	○	●	○	●	○
○	●	●	○	●	○	○	●
○	●	○	●	●	○	○	●
○	●	○	●	○	●	●	○
○	●	○	●	○	●	○	●

Corollaries

- In a real game of Mastermind we would *know* that our queries are satisfiable. Can we use this information to compute the size of the solution space?
  - In general, is it easier to compute the number of solutions, knowing that they are at least k?

Corollaries

- In a real game of Mastermind we would *know* that our queries are satisfiable. Can we use this information to compute the size of the solution space?
  - In general, is it easier to compute the number of solutions, knowing that they are at least k?
- Let #MATCH be the problem of counting all the matchings (perfect and imperfect) in a given graph.

Lemma (Valiant, 1979)

#MATCH is #P-complete under Turing reductions.

Corollaries

- In a real game of Mastermind we would *know* that our queries are satisfiable. Can we use this information to compute the size of the solution space?
  - In general, is it easier to compute the number of solutions, knowing that they are at least k?
- Let #MATCH be the problem of counting all the matchings (perfect and imperfect) in a given graph.

Lemma (Valiant, 1979)

#MATCH is #P-complete under Turing reductions.

- Then $\#\mathrm{SAT} \leqslant_T \#\mathrm{MATCH} \leqslant_{pars} \#\mathrm{MSP}$.
- The graphs with fewer than k edges are solved by hand; the others (which have at least k matchings) are mapped to #MSP.

Corollary

#MSP restricted to instances with at least k solutions is #P-hard.

Corollaries

Problem (Stuckman–Zhang, 2005)

Can we detect MSP instances with a unique solution?

Corollaries

Problem (Stuckman–Zhang, 2005)

Can we detect MSP instances with a unique solution?

- If we are *given* $k \geqslant 1$ solutions, can we tell if there are more?

Corollaries

Problem (Stuckman–Zhang, 2005)

Can we detect MSP instances with a unique solution?

- If we are *given* $k \geqslant 1$ solutions, can we tell if there are more?

Corollary

No, it is NP-complete.

- Not only *solving* Mastermind puzzles is hard, but *designing* puzzles around a solution is hard.

Corollaries

Problem (Stuckman–Zhang, 2005)

Can we detect MSP instances with a unique solution?

- If we are *given* $k \geqslant 1$ solutions, can we tell if there are more?

Corollary

No, it is NP-complete.

- Not only *solving* Mastermind puzzles is hard, but *designing* puzzles around a solution is hard.
- What if we *know* that the solution is unique? Can we find it?

Corollaries

Problem (Stuckman–Zhang, 2005)

Can we detect MSP instances with a unique solution?

- If we are *given* $k \geqslant 1$ solutions, can we tell if there are more?

Corollary

No, it is NP-complete.

- Not only *solving* Mastermind puzzles is hard, but *designing* puzzles around a solution is hard.
- What if we *know* that the solution is unique? Can we find it?

Corollary

No, it is NP-hard under randomized Turing reductions.

Open problems

- $\#c$-MSP-WHITE $\in$ FP when c is a constant.
- $\#\sqrt[k]{n}$ -MSP-WHITE is $\#$P-complete for every $k \geqslant 1$.

Problem

What is the lowest order of growth of $c(n)$ such that $\#c(n)$-MSP-WHITE is $\#$P-complete?

Open problems

- Solving MSP is a sub-step of several heuristics.
  - But is it really necessary?

MASTERMIND

Input: (n, c, Q, k).
Output: YES *if the codebreaker has a strategy to guess the code within* k *attempts, given the set of queries* Q*.* NO *otherwise.*

Open problems

- Solving MSP is a sub-step of several heuristics.
  - But is it really necessary?

MASTERMIND

Input: (n, c, Q, k).
Output: YES *if the codebreaker has a strategy to guess the code within* k *attempts, given the set of queries* Q. NO *otherwise.*

- MASTERMIND $\in$ PSPACE, due to Chvátal's strategy.

Problem

Is MASTERMIND PSPACE-complete?