**Approximation Algorithms and Hardness of Approximation** April 26-30, 2013

# Lecture 17-18

*Lecturer: Ola Svensson* *Scribes: Rajai Nasser*

We continue the discussion of the PCP theorem and its close relation to the hardness of approximation of some interesting problems such as MAX3-SAT. In the previous lecture we have seen that there are two points of view to see the PCP theorem: the first one characterizes the NP problems by stating that NP = PCP($\log n, 1$), the other one states that it is NP-hard to approximate the MAX3-SAT problem with an approximation guarantee that is arbitrarily close to 1 (i.e., there is no PTAS for the MAX3-SAT problem). In this note, we will see that the hardness of approximation point of view can be stated using any NP-complete problem qCSPw (which will be defined later).

In section 1, we prove a weak version of the PCP theorem which states that NP $\subset$ PCP(Poly($n$), 1). The main steps of the proof of the PCP theorem are then explained in section 2.

## 1 A weak version of the PCP theorem

### 1.1 Walsh-Hadamard codes

In the previous lecture, we have introduced the Walsh-Hadamard codes. We will use these codes and their nice properties in order to prove a weak version of the PCP theorem (NP $\subset$ PCP(Poly($n$), 1)).

Recall that a Walsh-Hadamard code is a mapping $WH : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^{2^n}$, which maps a string $u \in \mathbb{F}_2^n$ to the truth table of the function $x \longrightarrow x \odot u := \sum_{i=1}^{n} x_i u_i$ ($\mathbb{F}_2$ denotes the Galois field $GF(2) := \{0, 1\}$). In other words, the codewords of the Walsh-Hadamard codes are the truth tables of all linear functions. We have seen in the previous lecture that the Walsh-Hadamard code has two interesting properties:

- **Local testability:** For any $\delta \in (0, \frac{1}{2})$, there is a *$\delta$-linearity test* which randomly reads $O(\frac{1}{\delta})$ bits of a $2^n$ string (which can be seen as the truth table of a function from $\mathbb{F}_2^n$ to $\mathbb{F}_2$), and rejects any function that is not $(1-\delta)$-close to a linear function with a probability of at least $\frac{1}{2}$. Recall that two functions $f, g : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2$ are said to be $(1-\epsilon)$-close if they disagree on at most $\epsilon$ fraction of their input.
- **Local decodability:** If $f$ is $(1-\delta)$-close to a linear function $\hat{f}$. We can compute $\hat{f}(x)$ for any $x \in \mathbb{F}_2^n$ with a probability of at least $1 - 2\delta$: we randomly choose $x'$ uniformly in $\mathbb{F}_2^n$ and then compute $f(x \oplus x') \oplus f(x')$.

A very interesting property of binary linear functions, which we extensively use in this section, is the random subsum property:

**Proposition 1** *(Random Subsum Property) For any $u, v \in \mathbb{F}_2^n$ such that $u \neq v$, and for half the possible values of $x \in \mathbb{F}_2^n$ we have $u \odot x \neq v \odot x$.*

**Proof** Note that $u \odot x \neq v \odot x$ if and only if $(u \odot x) \oplus (v \odot x) = (u \oplus v) \odot x = 1$. Let $w = u \oplus v$ which is different from 0 since $u \neq v$. We have $w \odot x = \sum_{i=1}^{n} w_i x_i = \sum_{i:w_i \neq 0} x_i$.

Let $X = (X_1, \ldots, X_n)$ be a uniform random vector in $\mathbb{F}_2^n$ (i.e., $X_1, \ldots, X_n$ are i.i.d. uniform Bernoulli random variables), we have $w \odot X = \sum_{i:w_i \neq 0} X_i$. Now since $w \neq 0$, there exists at least one index $i$ such that $w_i \neq 0$. Therefore, $w \odot X = \sum_{i:w_i \neq 0} X_i$ is a uniform Bernoulli random variable, and $w \odot X = 1$ with probability $\frac{1}{2}$. Therefore, $w \odot x = 1$ (i.e., $u \odot x \neq v \odot x$) for half the possible values of $x \in \mathbb{F}_2^n$. ■

## 1.2 The QUADEQ problem

In order to prove that NP $\subset$ PCP(Poly($n$),1), it is sufficient to choose any NP-complete problem A, and prove that A $\in$ PCP(Poly($n$),1): If B is any NP problem, we can obtain a PCP(Poly($n$),1) verifier for B by reducing it to A and then apply the PCP(Poly($n$),1) verifier of A. Therefore, we have to choose an NP-complete problem A for which it is easy to prove that A $\in$ PCP(Poly($n$),1). It turns out that QUADEQ (defined below) is a good candidate.

**Definition 2** *In the* QUADEQ *problem, we are given a system of $m$ quadratic equations in $n$ variables in $\mathbb{F}_2$, and we have to decide whether there exists an assignment of the $n$ variables that satisfies all the quadratic equations.*

**Example 1** *Here is an example of a system of quadratic equations:*

$$\begin{aligned} u_1u_2 + u_3u_1 + u_2u_3 &= 1 \\ u_2u_2 + \quad u_1 &= 0 \\ u_3u_2 + u_2u_1 + u_1u_1 &= 1 \end{aligned}$$

*It is easy to see that the assignment $u_1 = u_2 = u_3 = 1$ satisfies all the three equations.*

**Theorem 3** QUADEQ *is* NP*-complete.*

Since $X^2 = X$ for any variable $X \in \mathbb{F}_2$, we can assume that all the terms in the quadratic equations are of degree 2. And since there are $n$ such variables, there are exactly $n^2$ possible terms of degree 2. Therefore, we can reformulate the QUADEQ problem as follows:

- We are given an $m \times n^2$ $\mathbb{F}_2$-matrix $A$, and an $m$-dimensional vector $b \in \mathbb{F}_2^m$.
- We have to decide whether there exists $U \in \mathbb{F}_2^{n^2}$ such that:
  - $AU = b$.
  - $U$ is equal to $u \otimes u$ (i.e., $U_{n(i-1)+j} = u_i u_j$ for $1 \le i, j \le n$), for some $u \in \mathbb{F}_2^n$. This condition is necessary in order to ensure that $U$ corresponds to the terms of degree 2 corresponding to a certain assignment.

## 1.3 PCP(Poly(n),1) verifier for QUADEQ

The first verifier that we can think of is a one that gets access to a string $(U, u) \in \mathbb{F}_2^{n^2} \times \mathbb{F}_2^n$ and accepts the string as a valid proof if $AU = b$ and $U = u \otimes u$. Clearly, if a QUADEQ instance is satisfiable by a certain assignment $u_1, \ldots, u_n$, then the verifier accepts the proof $(U, u)$, where $u = (u_1, \ldots, u_n)$ and $U = u \otimes u$. On the other hand, if a QUADEQ instance is not satisfiable, then the verifier rejects any string $(U, u)$. This shows that the suggested verifier is an NP verifier for the QUADEQ problem.

Unfortunately, the proof $(U, u)$ is not robust enough in order to use it in a PCP(Poly($n$),1) verifier. Remember that a PCP(Poly($n$),1) verifier reads Poly($n$) random bits based on which (and on the input) it reads a constant number of bits from the proof, and then it finally decides whether or not to accept the proof as a valid one. Therefore, since we are basing our decision on a constant number of bits of the proof, we have to robustify it. One way to do this is by requiring that a large portion of the proof has to depend on a large number of bits of the original proof $(U, u)$, so that a constant number of bits that is randomly chosen from the proof will be likely to be "representative" of the whole original proof $(U, u)$.

Walsh-Hadamard codes is an excellent tool that can help us achieve this requirement. Therefore, instead of getting access to an $(n^2 + n)$-long string $(U, u)$, our PCP verifier will get access to a $(2^{n^2} + 2^n)$-long string $(g, f) \in \mathbb{F}_2^{2^{n^2}} \times \mathbb{F}_2^{2^n}$ ($g$, resp. $f$, can be thought of as the truth table a function from $\mathbb{F}_2^{n^2}$, resp. from $\mathbb{F}_2^n$, to $\mathbb{F}_2$, we will identify $g$ and $f$ with those functions). A valid proof will consist of a pair of Walsh-Hadamard codes $g = WH(U)$ and $f = WH(u)$, where $(U, u) \in \mathbb{F}_2^{n^2} \times \mathbb{F}_2^n$ satisfies $AU = b$ and $U = u \otimes u$. Thus, the PCP verifier will check $(g, f)$ in 3 steps:

- Step 1: Check that $f$ and $g$ are codewords (i.e., linear functions).
- Step 2: Check that $g = WH(u \otimes u)$, where $f = WH(u)$.
- Step 3: Check that $U = u \otimes u$ satisfies $AU = b$.

Of course, we have to do these steps by making only a constant number of random queries from $(f, g)$, and we have to do it in such a way that the verifier accepts a valid proof with probability 1, and if the problem instance is not satisfiable then the verifier must reject any proof with a probability of at least $\frac{1}{2}$.

## Step 1

From the local testability property of Walsh-Hadamard codes, there is a 99%-linearity test which uses only a constant number $K = O(\frac{1}{0.01})$ of queries from a function, accepts any linear function, and rejects with probability of at least $\frac{1}{2}$ any function that is not 99%-close to a linear function. Step 1 applies this linearity test twice to each of $f$ and $g$. We have the following:

- If $f$ and $g$ are linear functions, $(f, g)$ passes Step 1 with probability 1.
- If either $f$ or $g$ is not 99%-close to a linear function, $(f, g)$ passes Step 1 with probability of at most $\left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2 = \frac{1}{2}$, i.e., $(f, g)$ is rejected with a probability of at least $\frac{1}{2}$.
- Step 1 makes $2K$ random queries from $f$ and $2K$ random queries from $g$. As the description of $f$ and $g$ contains $2^n$ and $2^{n^2}$ bits respectively, we need $2K(n^2 + n)$ random bits for Step 1.

## Step 2

Suppose that $f$ and $g$ are 99%-close to linear functions $\hat{f}$ and $\hat{g}$ respectively, and suppose that $\hat{g} = WH(U)$ and $\hat{f} = WH(u)$.

**Claim 4** *If $U = u \otimes u$, then $\hat{g}(r \otimes r') = \hat{f}(r) \cdot \hat{f}(r')$ for any $r, r' \in \mathcal{F}_2^n$.*

**Proof**

$$\begin{aligned}
\hat{f}(r) \cdot \hat{f}(r') = (u \odot r) \cdot (u \odot r') &= \Big(\sum_{i=1}^{n} u_i r_i\Big)\Big(\sum_{j=1}^{n} u_j r'_j\Big) = \sum_{1 \le i,j \le n} u_i u_j r_i r'_j \\
&= \sum_{1 \le i,j \le n} (u \otimes u)_{i(n-1)+j} (r \otimes r')_{i(n-1)+j} = \sum_{l=1}^{n^2} (u \otimes u)_l (r \otimes r')_l \\
&= (u \otimes u) \odot (r \otimes r') = \hat{g}(r \otimes r')
\end{aligned}$$

■

**Claim 5** *If $U \neq u \otimes u'$, then for at least $\frac{1}{4}$ of the possible values of $(r, r') \in \mathbb{F}_2^n \times \mathbb{F}_2^n$, we have $\hat{f}(r) \cdot \hat{f}(r') \neq \hat{g}(r \otimes r')$.*

**Proof** Let $\hat{U}$ be the $n \times n$ matrix defined by $\hat{U}_{ij} = U_{(n-1)i+j}$. We have:

- Since $U \neq u \otimes u$, there exists at least two indices $i$ and $j$ for which we have $U_{n(i-1)+j} \neq u_i u_j$. Therefore, $\hat{U}_{ij} \neq (uu^t)_{ij}$ for these indices $i$ and $j$. Thus, $\hat{U} \neq uu^t$.
- $g(r \otimes r') = \sum_{1 \leq i,j \leq n} U_{(n-1)i+j}(r \otimes r')_{(n-1)i+j} = \sum_{1 \leq i,j \leq n} \hat{U}_{ij} r_i r'_j = \sum_{1 \leq i,j \leq n} r_i \hat{U}_{ij} r'_j = r^t \hat{U} r'$.
- $f(r) \cdot f(r') = \Big(\sum_{i=1}^{n} r_i u_i\Big)\Big(\sum_{j=1}^{n} r'_j u_j\Big) = (r^t u)(u^t r') = r^t (uu^t) r'$.

Since $\hat{U} \neq uu^t$, there exists at least one column $c_1$ of $\hat{U}$, and one column $c_2$ of $uu^t$, which have the same position in $\hat{U}$ and $uu^t$ respectively, and for which we have $c_1 \neq c_2$. By the random subsum property we have $r^t c_1 = r \odot c_1 \neq r \odot c_2 = r^t c_2$ for exactly one half of the possible values of $r \in \mathbb{F}_2^n$. For those values of $r$, we must have $r^t \hat{U} \neq r^t(uu^t)$ since $r^t c_1 \neq r^t c_2$. Now fix a value of $r$ satisfying $r^t \hat{U} \neq r^t(uu^t)$, by applying the random subsum property for the two vectors $r^t \hat{U}$ and $r^t(uu^t)$ we conclude that for half the possible values of $r' \in \mathbb{F}_2^n$ we have $r^t \hat{U} r' \neq r^t(uu^t) r'$.

We conclude that for at least one fourth of the possible values of $(r, r') \in \mathbb{F}_2^n \times \mathbb{F}_2^n$, we have

$$\hat{f}(r) \cdot \hat{f}(r') = r^t(uu^t)r' \neq r^t \hat{U} r' = \hat{g}(r \otimes r').$$

Note that the fraction of the possible values of $(r, r')$ for which we have $\hat{f}(r) \cdot \hat{f}(r') \neq \hat{g}(r \otimes r')$ might be more than one fourth, as the matrices $\hat{U}$ and $uu^t$ might differ in more than one column. ■

Motivated by the previous two claims, Step 2 of the verifier will repeat the following 3 times:

- Choose $r \in \mathbb{F}_2^n$ and $r' \in \mathbb{F}_2^n$ uniformly and independently.
- Using local decodability, decode $\hat{f}(r)$, $\hat{f}(r')$ and $\hat{g}(r \otimes r')$.
- Check if $\hat{f}(r) \cdot \hat{f}(r') = \hat{g}(r \otimes r')$.

We have the following:

- If $f = \hat{f} = WH(u)$ and $g = \hat{g} = WH(u \otimes u)$, then according to claim 4, $(f, g)$ passes Step 2 with probability 1.
- If $f$ and $g$ are 99%-close to two linear function $\hat{f} = WH(u)$ and $\hat{g} = WH(U)$ respectively, such that $U \neq u \otimes u$, then:
  - For each iteration, each of $\hat{f}(r)$, $\hat{f}(r')$ and $\hat{g}(r \otimes r')$ is incorrectly decoded with a probability of at most 2%. Therefore, all of the three are correctly decoded with a probability of at least 94%.
  - Given that $\hat{f}(r)$, $\hat{f}(r')$ and $\hat{g}(r \otimes r')$ are correctly decoded in a particular iteration, the probability that $(f, g)$ is rejected in this iteration is at least $\frac{1}{4}$ according to claim 5.
  - Therefore, in each iteration the probability that $(f, g)$ is rejected is at least $\frac{0.94}{4}$. Thus, the probability that $(f, g)$ is rejected in at least one iteration is at least $1 - \big(1 - \frac{0.94}{4}\big)^3 > \frac{1}{2}$.
- In each iteration, we need $n$ random bits for the choice of each of $r$ and $r'$. Moreover, in order to decode $\hat{f}(r)$, $\hat{f}(r')$ and $\hat{g}(r \otimes r')$, we need $n$, $n$ and $n^2$ random bits respectively, and we query two bits for each one of them (see the previous lecture to see how the local decodability is performed). Therefore, Step 2 uses a total of $3(4n + n^2) = 12n + 3n^2$ random bits, and queries a total of $3(2 + 2 + 2) = 18$ bits from $(f, g)$.

**Step 3**

Suppose now that $f$ and $g$ are 99%-close to the linear function $\hat{f}$ and $\hat{g}$, and suppose that $\hat{g} = WH(U)$ and $\hat{f} = WH(u)$, where $U = u \otimes u$.

**Claim 6** *Let $y$ be a random vector uniformly chosen in $\mathbb{F}_2^m$. If $AU = b$ then $y^t AU = y^t b$ with probability 1. On the other hand, if $AU \neq b$ then $y^t AU \neq y^t b$ with probability $\frac{1}{2}$.*

**Proof** The claim is trivial for $AU = b$. We simply apply the random subsum property for the case where $AU \neq b$. ■

Notice that $y^t AU = (A^t y)^t U = (A^t y) \odot U = \hat{g}(A^t y)$ and so $y^t AU$ can be obtained by decoding $\hat{g}(A^t y)$. Motivated by the previous claim, Step 3 repeats the following twice:

- Choose $y$ uniformly in $\mathbb{F}_2^m$.
- Decode $\hat{g}(A^t y)$.
- Check if $\hat{g}(A^t y) = y^t b$.

We have the following:

- If $g = \hat{g} = WH(U)$ and $AU = b$, then $g(A^t y) = y^t b$ for all $y$ and $(f, g)$ passes Step 3 with probability 1.
- If $g$ is 99% close to a linear function $\hat{g} = WH(U)$ and $AU \neq b$, then:
  - In each iteration there is a 98% probability that $\hat{g}(A^t y)$ will be decoded correctly.
  - Given that $\hat{g}(A^t y)$ is decoded correctly in one iteration, there is a $\frac{1}{2}$ probability that $(f, g)$ will be rejected by this iteration.
  - Therefore, $(f, g)$ will be rejected by Step 3 with a probability of at least $1 - \left(1 - \frac{0.98}{2}\right)^2 > \frac{1}{2}$.
- Each iteration requires $m$ random bits in order to choose $y \in \mathbb{F}_2^m$, and we need $n^2$ additional random bits in order to decode $\hat{g}(A^t y)$. Moreover, two queries from $g$ is needed in order to decode $\hat{g}(A^t y)$. Therefore, Step 3 uses a total of $2(m + n^2) = 2m + 2n^2$ random bits and queries a total of $2(2) = 4$ bits from $g$.

**Completeness of the proposed PCP verifier**

Suppose that the instance of the QUADEQ problem is satisfiable, there must exist a vector $u \in \mathbb{F}_2^n$ such that $A(u \otimes u) = b$. If we let $f = WH(u)$ and $g = WH(u \otimes u)$, the proof $(f, g)$ will pass all the three steps of the verifier with probability 1.

**Soundness of the proposed PCP verifier**

Suppose that the instance of the QUADEQ problem is not satisfiable, then for any vector $u \in \mathbb{F}_2^n$ we must have $A \cdot (u \otimes u) \neq b$. Let $(f, g)$ be any $(2^n + 2^{n^2})$-long string in $\mathbb{F}_2^{2^n} \times \mathbb{F}_2^{2^{n^2}}$. We have:

- If either $f$ or $g$ is not 99%-close to a linear function, then $(f, g)$ will be rejected by Step 1 with a probability of at least $\frac{1}{2}$.
- If both $f$ and $g$ are 99%-close to linear functions $\hat{f} = WH(u)$ and $\hat{g} = WH(U)$ respectively with $U \neq u \otimes u$, then $(f, g)$ will be rejected by Step 2 with a probability of at least $\frac{1}{2}$.

- If both $f$ and $g$ are 99%-close to linear functions $\hat{f} = WH(u)$ and $\hat{g} = WH(U)$ respectively with $U = u \otimes u$, then we must have $AU \neq b$. Therefore, $(f, g)$ will be rejected by Step 3 with a probability of at least $\frac{1}{2}$.
- We conclude that in all cases, there is a probability of at least $\frac{1}{2}$ that $(f, g)$ will be rejected by at least one step of the verifier.
- The total number of random bits that are used is $(2K+5)n^2+(2K+12)n+2m$ which is polynomial in the size of the input.
- The total number of queries from the proof $(f, g)$ is $4K + 22$ which is constant.

This shows that QAUDEQ $\in$ PCP(Poly($n$,1)), which proves that NP $\subset$ PCP(Poly($n$,1)).

## 2 PCP theorem

In this section we will start the proof of the PCP theorem. We will first prove the hardness of approximation point of view of it, and then we will prove that NP $\subset$ PCP$(\log n, 1)$.

### 2.1 CSP problems

**Definition 7** *A constraint $C$ of arity $q$ and of alphabet size $w$ on the variables $X_1, X_2, \ldots, X_n$ is a $(q+1)$-tuple $C := (f_C, j_{C,1}, \ldots, j_{C,q})$, where $f_C$ is a mapping from $[w]^q$ to $\{0, 1\}$, $[w] := \{0, \ldots, w-1\}$ and $1 \leq j_{C_1}, \ldots, j_{C_q} \leq n$. An assignment of $X_1, X_2, \ldots, X_n$ is said to* satisfy *$C$ if and only if $f_C(X_{j_{C,1}}, \ldots, X_{j_{C,1}}) = 1$. We say that $C$ is a constraint on the $q$ variables $X_{j_{C,1}}, \ldots, X_{j_{C,1}}$ to indicate that $j_{C_1}, \ldots, j_{C_q}$ are the indices that appear in $C$. We will also write $C(X_{j_{C,1}}, \ldots, X_{j_{C,1}})$ in order to denote $f_C(X_{j_{C,1}}, \ldots, X_{j_{C,1}})$.*

**Definition 8** *Let $q \geq 2$ and $w \geq 2$ be two integers. In the* constraint satisfaction problem *with arity $q$ and alphabet size $w$, which we denote by $q\mathrm{CSP}_w$, we are given a set of $m$ constraints $C_1, \ldots, C_m$ of arity $q$ and of alphabet size $w$ on the $n$ variables $X_1$, …,$X_n$. We have to decide whether there exists an assignment of the variables $X_1, \ldots, X_n$ that satisfies all the $m$ constraints.*

**Proposition 9** *If $q > 2$ or $w > 2$, then $q\mathrm{CSP}_w$ is* NP*-complete.*

**Proof** It is easy to see that $q\mathrm{CSP}_w$ is NP for any $q$ and $w$ (consider any assignment as a proof, it is possible to check it in polynomial time).

Since the disjunction is a particular constraint, there is a trivial polynomial time reduction from $q$SAT to $q\mathrm{CSP}_w$ for any $q > 2$ and any $w \geq 2$. Since $q$SAT is NP-complete for $q > 2$, $q\mathrm{CSP}_w$ is NP-complete if $q > 2$.

If $q = 2$ and $w \geq 3$, there is a polynomial time reduction from the $w$-colorability problem to $2\mathrm{CSP}_w$: We consider a variable $X_v$ for each vertex $v$ and we add a constraint $C_e$ that is satisfied if and only if $X_u \neq X_v$ for each edge $e = uv$. Since the $w$-colorability problem is known to be NP-complete for $w > 2$, $q\mathrm{CSP}_w$ is NP-complete if $w > 2$. ■

**Definition 10** *For any $q\mathrm{CSP}_w$ instance $\varphi$, the* value *of the instance $\varphi$, denoted* $\mathrm{Val}(\varphi)$ *is the maximal possible fraction of satisfiable constraints in $\varphi$. We clearly have* $\mathrm{Val}(\varphi) = 1$ *if $\varphi$ is satisfiable, and* $\mathrm{Val}(\varphi) \leq 1 - \frac{1}{m}$ *otherwise.*

**Definition 11** *Let $q \geq 2$ and $w \geq 2$ be two integers. In the* maximum constraint satisfaction problem, *which is denoted by* MAX$q\mathrm{CSP}_w$, *we are given an instance $\varphi$ of $q\mathrm{CSP}_w$ and we are asked to determine* $\mathrm{Val}(\varphi)$.

## 2.2 Complete linear reduction

**Definition 12** *A complete linear (CL) reduction from a problem A to a problem B, is a mapping f from the instances of A to the instances of B that is computable in polynomial time and which is:*

- *Complete: If $\varphi$ is an instance of A, then $\varphi$ is satisfiable if and only if $f(\varphi)$ is satisfiable.*
- *Linear: The size of the input in $f(\varphi)$ is linear in the size of $\varphi$, i.e., there exists a constant C such that for any instance of A we have $SIZE(f(\varphi)) \leq C.SIZE(\varphi)$. C is called the linear constant of the CL-reduction.*

Clearly, the composition of two CL-reductions is also a CL-reduction.

## 2.3 Gap amplifying mapping

**Definition 13** *An $(\alpha, \epsilon_0)$-gap amplifying mapping from $q_1\mathrm{CSP}_{w_1}$ to $q_2\mathrm{CSP}_{w_2}$ is a CL-reduction f from $q_1\mathrm{CSP}_{w_1}$ to $q_2\mathrm{CSP}_{w_2}$ such that for any $0 \leq \epsilon \leq \epsilon_0$ and any instance $\varphi$ of $q_1\mathrm{CSP}_{w_1}$ we have:*

$$\mathrm{Val}(f(\varphi)) \leq \max\{1 - \alpha(1 - \mathrm{Val}(\varphi)), 1 - \epsilon_0\}.$$

*In other words for any $\epsilon < \frac{\epsilon_0}{\alpha}$ we have:*

$$\mathrm{Val}(\varphi) \leq 1 - \epsilon \Rightarrow \mathrm{Val}(f(\varphi)) \leq 1 - \alpha\epsilon.$$

*$\alpha$ is called an* amplification factor *of f.*

We can easily see why such a reduction is called a gap amplifying mapping: Since the minimum number of non satisfiable constraints $(1 - \mathrm{Val}(\varphi))$ is either amplified by a factor of $\alpha$ (i.e., $1 - \mathrm{Val}(f(\varphi)) = \alpha(1 - \mathrm{Val}(\varphi))$), or becomes higher than a constant threshold (i.e., $1 - \mathrm{Val}(\varphi) \geq \epsilon_0$).

**Lemma 14** *The composition of an $(\alpha, \epsilon_0)$-gap amplifying mapping f with another $(\alpha', \epsilon_0')$-gap amplifying mapping g, is an $(\alpha'\alpha, \min\{\epsilon_0', \alpha'\epsilon_0\})$-gap amplifying mapping. Moreover, if $\alpha' \geq 1$, $g \circ f$ is an $(\alpha'\alpha, \min\{\epsilon_0', \epsilon_0\})$-gap amplifying mapping.*

**Proof** Trivial. ■

## 2.4 Key lemma and proof of the PCP theorem

The key lemma that is used to prove the PCP theorem is the following:

**Lemma 15 (Key lemma)** *There exists an integer $q_0 \geq 3$ and a real number $0 < \epsilon_0 < 1$, for which there exists a $(2, \epsilon_0)$-gap amplifying mapping from $q_0\mathrm{CSP}_2$ to $q_0\mathrm{CSP}_2$.*

It is not hard to see how the key lemma can be used to prove the PCP theorem. In fact, it leads to a direct proof of the hardness of approximation point of view of the PCP theorem:

**Theorem 16** (Hardness of approximation point of view of the PCP theorem) *There exists an integer $q_0 > 2$ and a real number $\epsilon_0 > 0$ such that it is* NP*-hard to approximate* $\mathrm{MAX}q_0\mathrm{CSP}_2$ *with an approximation guarantee that is strictly better than $1 - \epsilon_0$ (i.e., there is no* PTAS *for* $\mathrm{MAX}q_0\mathrm{CSP}_2$*).*

**Proof** Let $\epsilon_0 > 0$ and $q_0 > 2$ be as in the key lemma, let $f$ be the $(2, \epsilon_0)$-gap amplifying mapping from $q_0\mathrm{CSP}_2$ to $q_0\mathrm{CSP}_2$ which is guaranteed to exist by the key lemma, and let $C$ be the linear constant of $f$. Let $\varphi$ be any instance of $q_0\mathrm{CSP}_2$ containing $m$ constraints. By applying the reduction $f$ to $\varphi$ $\lceil \log_2 m \rceil$ times, we obtain an instance $f^{\lceil \log_2 m \rceil}(\varphi) \in q_0\mathrm{CSP}_2$, whose size is

$$SIZE(f^{\lceil \log_2 m \rceil}(\varphi)) \leq C^{\lceil \log_2 m \rceil} SIZE(\varphi) \leq C^{\log_2 m + 1} SIZE(\varphi) = C m^{\log_2 C} SIZE(\varphi)$$

which is polynomial in the size of $\varphi$. Moreover, lemma 14 implies that $f^{\lceil \log_2 m \rceil}$ is a $(2^{\lceil \log_2 m \rceil}, \epsilon_0)$-gap amplifying mapping. Therefore,

- If $\varphi$ is satisfiable, $f^{\lceil \log_2 m \rceil}(\varphi)$ is satisfiable as well (i.e., $\mathrm{Val}(f^{\lceil \log_2 m \rceil}(\varphi)) = \mathrm{Val}(\varphi) = 1$) since $f$ is a CL-reduction.
- If $\varphi$ is not satisfiable, $\mathrm{Val}(\varphi) \leq 1 - \frac{1}{m}$. Therefore, we have

$$\begin{aligned}\mathrm{Val}(f^{\lceil \log_2 m \rceil}(\varphi)) &\leq \max\{1 - 2^{\lceil \log_2 m \rceil}(1 - \mathrm{Val}(\varphi)), 1 - \epsilon_0\} \leq \max\left\{1 - m\Big(1 - \Big(1 - \frac{1}{m}\Big)\Big), 1 - \epsilon_0\right\} \\ &= \max\{0, 1 - \epsilon_0\} = 1 - \epsilon_0\end{aligned}$$

This shows that it is NP-hard to approximate MAX$q_0$CSP$_2$ with an approximation guarantee that is strictly better than $1 - \epsilon_0$ as $q_0$CSP$_2$ can be reduced in polynomial time to any such an approximation algorithm: after computing $f^{\lceil \log_2 m \rceil}(\varphi)$ in polynomial time, we can use the approximation algorithm to distinguish whether we have $\mathrm{Val}(f^{\lceil \log_2 m \rceil}(\varphi)) = 1$ or $\mathrm{Val}(f^{\lceil \log_2 m \rceil}(\varphi)) \leq 1 - \epsilon_0$, which is equivalent to distinguishing whether $\varphi$ is satisfiable or no. ■

**Theorem 17** (PCP Theorem) NP $\subset$ PCP($\log n$, 1)

**Proof** Let $\epsilon_0 > 0$ and $q_0 > 2$ be as in the key lemma, and let $f$ be the $(2, \epsilon_0)$-gap amplifying mapping from $q_0$CSP$_2$ to $q_0$CSP$_2$ which is guaranteed to exist by the key lemma. Let $\psi$ be an instance of an NP-problem A whose size is $n$. Since $q_0$CSP$_2$ is NP-complete, it is possible to reduce $\psi$ to an instance $\varphi$ of $q_0$CSP$_2$ in polynomial time. Let $m$ be the number of constraints in $\varphi$, let $\varphi' = f^{\lceil \log_2 m \rceil}(\varphi)$, let $n'$ be the number of variables in $\varphi'$ and let $m'$ be the number of constraints in $\varphi'$. Note that both $n'$ and $m'$ are polynomial in $n$ (the size of $\psi$).

We construct a PCP($\log n$,1)) verifier for the problem $A$ as follows:

- The verifier gets access to a string of $\lceil \frac{-1}{\log_2(1-\epsilon_0)} \rceil \log m' = O(\log n)$ random bits. The random string is interpreted as a sequence of $l := \lceil \frac{-1}{\log_2(1-\epsilon_0)} \rceil$ indices $i_1, \ldots, i_l$ taking values uniformly and independently in $\{1, \ldots, m'\}$.
- A proof consists of a string of $n'$ bits. The proof is interpreted as an assignment of the $n'$ variables of $\varphi'$.
- The verifier reduces $\psi$ to $\varphi$ and then to $\varphi'$ in polynomial time. Let $C_1, \ldots, C_{m'}$ be the $m'$ constraints of $\varphi'$.
- For $1 \leq j \leq l$, the verifier reads the $q_0$ bits corresponding to the $q_0$ variables appearing in $C_{i_j}$. Therefore, the verifier reads at most $q_0 l = q_0 \lceil \frac{-1}{\log_2(1-\epsilon_0)} \rceil = O(1)$ bits from the proof.
- For $1 \leq j \leq l$, the verifier checks if $C_{i_j}$ is satisfied by the bits that are read from the proof.

We have the following:

- Completeness: If $\psi$ is satisfiable, both $\varphi$ and $\varphi'$ are satisfiable. Therefore, there exists an assignment that satisfies all the constraints of $\varphi'$. For such an assignment as a proof, our verifier accepts it with probability 1.
- Soundness: If $\psi$ is not satisfiable, $\varphi$ is not satisfiable and $\mathrm{Val}(\varphi') \leq 1 - \epsilon_0$ (see the proof of the previous theorem). Therefore, the maximal possible fraction of satisfiable constraints of $\varphi'$ is at most $1 - \epsilon_0$. Now consider any assignment of the variables of $\varphi'$, the number of satisfied constraints in this assignment is at most $(1 - \epsilon_0)m'$, and the probability that a randomly chosen constraint is satisfied is at most $(1 - \epsilon_0)$. Therefore, the probability that our verifier accepts a proof is at most $(1 - \epsilon_0)^l = (1 - \epsilon_0)^{\lceil \frac{-1}{\log_2(1-\epsilon_0)} \rceil} \leq (1 - \epsilon_0)^{\frac{-1}{\log_2(1-\epsilon_0)}} = 2^{\log_2(1-\epsilon_0)\frac{-1}{\log_2(1-\epsilon_0)}} = 2^{-1}$. Therefore, if $\psi$ is not satisfiable then any proof is rejected with a probability of at least $\frac{1}{2}$.

This shows that there is a PCP($\log n$,1)) verifier for any NP problem. Therefore, NP $\subset$ PCP$(\log n, 1)$. ■

## 2.5 Dinur's gap amplification lemma

The proofs of the last two theorems show that it is sufficient to prove the key lemma in order to prove the PCP theorem. The key lemma can be proved in two steps:

1. Gap amplification by Dinur's lemma: There exist two integers $q_0 > 2$ and $w > 2$, and a real number $0 < \epsilon_1 < 1$, such that there exists a $(6, \epsilon_1)$-gap amplifying mapping from $q_0\text{CSP}_2$ to $2\text{CSP}_w$.
2. Alphabet reduction: There exists a real number $0 < \epsilon_2 < 1$, for which there exists a $(\frac{1}{3}, \epsilon_2)$-gap amplifying mapping from $2\text{CSP}_w$ back to $q_0\text{CSP}_2$.

The composition of the above two gap amplifying mapping gives a $(2, \epsilon_0)$-gap amplifying mapping, where $\epsilon_0 = \min\{\epsilon_2, \frac{1}{3}\epsilon_1\}$ (see lemma 14), proving the key lemma.

The amplification factor that was used above for gap amplification is 6. In general, Dinur's lemma can guarantee any amplification factor:

**Lemma 18 (Dinur's gap amplification lemma - 2006)** *For every $l > 0$ and every integer $q \geq 2$, there exists an integer $w$ and a real number $0 < \epsilon_1 < 1$ depending only on $l$ and $q$, such that there exists an $(l, \epsilon_1)$-gap amplifying mapping from $q\text{CSP}_2$ to $2\text{CSP}_w$.*

The rest of this lecture and the next lecture will be dedicated for the proof of Dinur's gap amplification lemma. The proof of the alphabet reduction step can be found in the third problem of the fourth assignment. Dinur's gap amplification lemma will be proved in several sub-steps:

i Arity reduction: For any $q \geq 2$, there exists a $(\frac{1}{q}, 1)$-gap amplifying mapping from $q\text{CSP}_2$ to $2\text{CSP}_{2^q}$.

ii Making nice instances: For any $w_1 \geq 2$, there exists a constant $0 < \beta(w_1) < 1$ depending only on $w_1$, for which there exists a $(\beta(w_1), 1)$-gap amplifying mapping from $2\text{CSP}_{w_1}$ to $2\text{CSP}_{w_1}$ which maps any instance of $2\text{CSP}_{w_1}$ to a nice instance of $2\text{CSP}_{w_1}$ (Nice instances will be defined below).

iii Gap amplification: For any $l' > 0$ and any $w_1 \geq 2$, there exists $0 < \epsilon_1 < 1$ and an integer $w \geq 2$ depending only on $l'$ and $w$, for which there exists an $(l', \epsilon_1)$-gap amplifying mapping from nice instances of $2\text{CSP}_{w_1}$ to instances of $2\text{CSP}_w$.

By composing the gap amplifying mappings obtained from the above three sub-steps for $w_1 = (2^q)$ and $l' = \frac{q}{\beta(2^q)}l$, we get an $(l, \epsilon_1)$-gap amplifying mapping from $q\text{CSP}_2$ to $2\text{CSP}_w$, proving Dinur's gap amplification lemma. The first two sub-steps will be proved in this lecture, the gap amplification sub-step will be proved in the next lecture.

Note that in the first two sub-steps, the gap experiences a reduction rather than an amplification, but this reduction is necessary in order to transform the problem instance to a nicely structured one, after which we can apply the gap amplification step which can compensate for all the previous gap reductions: while it is not easy to amplify the gap of instances that are not nicely structured, it is possible to arbitrarily amplify the gap of nicely structured instances.

## 2.6 Arity reduction

**Lemma 19** *For any $q \geq 2$, there exists a $(\frac{1}{q}, 1)$-gap amplifying mapping from $q\text{CSP}_2$ to $2\text{CSP}_{2^q}$.*

**Proof** Let $\varphi$ be an instance of $q\mathrm{CSP}_2$ containing $n$ variables $U_1, \ldots, U_n$ and $m$ constraints $C_1, \ldots, C_m$. From $\varphi$ we construct an instance $\psi$ of $2\mathrm{CSP}_{2^q}$ as follows:

- $\psi$ has variables $U_1, \ldots, U_n$ and $m$ variables $Y_1, \ldots, Y_m$ taking values in $[2^q]$.
- $U_1, \ldots, U_n$ will be interpreted as the same variables of $\varphi$.
- For $1 \leq i \leq m$, the $q$ bits of $Y_i$ will be interpreted as the assignment of the $q$ variables $U_{j_{i,1}}, \ldots, U_{j_{i,q}}$ appearing in $C_i$.
- For $1 \leq i \leq m$ and $1 \leq k \leq q$, we add a constraint $C'_{i,k}$ on the 2 variables $Y_i$ and $U_{j_{i,k}}$ which is satisfied if and only if:
  - $U_{j_{i,k}} \in \{0, 1\}$.
  - $Y_i$ agrees with $U_{j_{i,k}}$, i.e., the $k^{th}$ bit of $Y_i$ is equal to $U_{j_{i,k}}$.
  - $Y_i$ corresponds to an assignment of $U_{j_{i,1}}, \ldots, U_{j_{i,q}}$ that satisfies $C_i$.

Let $f$ be the mapping that maps a $q\mathrm{CSP}_2$ instance $\varphi$ to a $2\mathrm{CSP}_{2^q}$ instance $\psi$ according to the above procedure. We have the following:

- The number of constraints of $\psi$ is $qm$, i.e., we have a linear increase in the size of the instance.
- It is easy to see that if $U_{j_{i,1}}, \ldots, U_{j_{i,q}}$ satisfies $C_i$, then $U_{j_{i,1}}, \ldots, U_{j_{i,q}}, Y_i$ satisfies $C'_{i,1}, \ldots, C'_{i,q}$, where $Y_i = \displaystyle\sum_{k=1}^{q} U_{j_{i,k}} 2^{k-1}$. On the other hand, if $U_{j_{i,1}}, \ldots, U_{j_{i,q}}, Y_i$ satisfies $C'_{i,1}, \ldots, C'_{i,q}$, then $U_{j_{i,1}}, \ldots, U_{j_{i,q}}$ satisfies $C_i$. Therefore, $\varphi$ is satisfiable if and only if $\psi$ is satisfiable. Thus, $f$ is a CL-reduction.
- Suppose that $\mathrm{Val}(\varphi) = 1 - \epsilon$, and consider any assignment of the variables $U_1, \ldots, U_n, Y_1, \ldots, Y_m$. Since $\mathrm{Val}(\varphi) = 1 - \epsilon$, there exists $r \geq \epsilon m$ constraints $C_{i_1}, \ldots, C_{i_r}$ that are unsatisfied. For each $1 \leq k \leq r$, at least one of the $q$ constraints $C'_{i_k,1}, \ldots, C'_{i_k,q}$ is not satisfied (otherwise, $C_{i_k}$ would be satisfied). Therefore, for any assignment of the variables of $\psi$, there are at least $\epsilon m$ unsatisfied constraints, and since there is a total of $qm$ constraints in $\psi$, we have:

$$\mathrm{Val}(\psi) \leq 1 - \frac{\epsilon}{q} = 1 - \frac{1}{q}\Big(1 - \mathrm{Val}(\varphi)\Big) = \max\Big\{1 - \frac{1}{q}\Big(1 - \mathrm{Val}(\varphi)\Big), 1 - 1\Big\}.$$

  This shows that $f$ is a $(\frac{1}{q}, 1)$-gap amplifying mapping from $q\mathrm{CSP}_2$ to $2\mathrm{CSP}_{2^q}$.

■

## 2.7 Nice instances

Before discussing the second sub-step (making nice instances), we have to define what is a nice instance. We first need to introduce a few notions.

**Definition 20** *Let $\varphi$ be an instance of* $2\mathrm{CSP}_w$. *The constraint graph $G(\varphi)$ of $\varphi$ is the graph whose vertices are the variables of $\varphi$ where there exists an edge between two vertices $U$ and $V$ for each constraint involving $U$ and $V$.*

Note that all the graphs considered here are in fact undirected multi-graphs (i.e., they might contain multiple edges as well as self-loops).

**Notation 1** *Let $G$ be a graph, and let $A$ be the adjacency matrix of $G$, we denote its second largest eigenvalue in absolute value by $\lambda(G)$.*

**Definition 21** *A $d$-regular graph $G$ is said to be $c$-expander if $\lambda(G) \leq c$. If $c \leq 0.9$, a $c$-expander is simply called an expander.*

**Lemma 22** *There exists an absolute constant $d''(c) \geq 2$ such that it is possible to construct a $d''(c)$-regular $c$-expander graph (in polynomial time) for any desired number of vertices. We denote $d''(0.9)$ by $d'$.*

Expander graphs will be a very important tool for proving the last two sub-step of Dinur's gap amplification lemma. They have the following nice property:

**Lemma 23** *Let $G = (V, E)$ be an expander. For any subset $S$ of $V$ such that $|S| \leq \frac{|V|}{2}$, if we choose an edge $uv$ uniformly from $E$ we get:*

$$\Pr(u \in S, v \in S) \leq \frac{|S|}{|V|}\Big(\frac{1+\lambda(G)}{2}\Big) \leq 0.95\frac{|S|}{|V|}$$

$$\Pr(u \in S, v \notin S) \geq \frac{|S|}{|V|}\Big(\frac{1-\lambda(G)}{2}\Big) \geq \frac{|S|}{20|V|}$$

Now we are ready to define the nice instances of $2\text{CSP}_w$:

**Definition 24** *A $2\text{CSP}_w$ instance $\varphi$ is said to be nice if and only if:*

- *$G(\varphi)$ is $d$-regular for some absolute constant $d$ (depending only on $w$).*
- *At any vertex of $G(\varphi)$, at least half of the edges are self-loops.*
- *$G(\varphi)$ is an expander.*

## 2.8 Making a nice instance

In this subsection, we will start the process of making a nice instance by making the constraint graph $d_1$-regular for a certain absolute constant $d_1 > 0$. Afterwards, we will complete the process of making a nice instance.

**Lemma 25** *There is an absolute constant $d_1 > 0$, such that for any $w_1 \geq 2$, there exists a constant $0 < \beta_1(w_1) < 1$ that depends only on $w_1$, for which there exists a $(\beta_1(w_1), 1)$-gap amplifying mapping from $2\text{CSP}_{w_1}$ to $2\text{CSP}_{w_1}$ which maps any instance of $2\text{CSP}_{w_1}$ to an instance of $2\text{CSP}_{w_1}$ whose constraint graph is $d_1 - regular$.*

**Proof** Let $d'$ be the absolute constant described in lemma 22, and let $d_1 = d'+1$. Let $\varphi$ be an instance of $2\text{CSP}_{w_1}$, let $U_1, \ldots, U_n$ be its variables and let $C_1, \ldots, C_m$ be its constraints. We will construct an instance $\psi$ of $2\text{CSP}_{w_1}$ whose constraint graph is $d_1$-regular as follows:

- We may assume that $G(\varphi)$ contains no self-loops: If $\varphi$ contains a constraint $C_j(U_i, U_i)$, we can replace $C_j$ by a constraint $B_j(U_i, U_l)$ for any $l \neq i$ which is satisfied if and only if $C_j(U_i, U_i) = 1$.
- For each variable $U_i$ that appears in $k_i$ constraints, let $e_{i,1}, \ldots, e_{i,k_i}$ be the $k_i$ edges that are incident to $U_i$. We add $k_i$ variables $Y_{i,e_{i,1}}, \ldots, Y_{i,e_{i,k_i}}$ to $\psi$. $Y_{i,e_{i,1}}, \ldots, Y_{i,e_{i,k_i}}$ are called the $Y_i$ variables.
- For each constraint $C_j$ of $\varphi$ on the two variables $U_i$ and $U_l$, let $e$ be the edge of $G(\varphi)$ corresponding to $C_j$. We add a constraint $C'_j$ on $Y_{i,e}$ and $Y_{l,e}$ to $\psi$, which is satisfied if and only if $(Y_{i,e}, Y_{l,e})$ satisfies $C_j$.

- For each variable $U_j$ that appears in $k$ constraints, use lemma 22 and construct a $d'$-regular graph $G_j$ on $k$-vertices. Associate each vertex of $G_i$ to one of the $Y_i$ variables, and for all edges $e \in G_i$ add an equality constraint between the two variables corresponding to the vertices incident to $e$.

It is easy to see the following:

- The total number of vertices in $G(\psi)$ is equal to the sum of degrees of the vertices of $G(\varphi)$, which is equal to twice the number of edges. Therefore, the total number of vertices in $G(\psi)$ is $2m$ which is linear in the size of $\varphi$.
- $G(\psi)$ is $(d'+1)$-regular. Therefore, the number of edges is equal to $\frac{d'+1}{2}(2m) = d_1 m$, i.e., the number of constraints in $\psi$ is equal to $d_1 m$ which is linear in the size of $\varphi$.
- $\varphi$ is satisfiable if and only if $\psi$ is satisfiable.

Let $f_1$ be the mapping that maps $2\mathrm{CSP}_w$ instances to $2\mathrm{CSP}_w$ instances whose constraint graphs are $d_1$-regular according to the above procedure. The above three points show that $f_1$ is a CL-reduction.

Now let $\varphi$ be an instance where $\mathrm{Val}(\varphi) = 1-\epsilon$, and let $\psi = f_1(\varphi)$. Now consider any assignment of the variables of $\psi$. For each $1 \le i \le n$, use the plurality rule and assign $U_i$ the value in $[w]$ that is the most common among the $Y_i$ variables. There are at least $\epsilon m$ unsatisfied $\varphi$-constraints for the plurality-rule assignment of $U_1, \ldots, U_n$. Let $t_i$ be the number of $Y_i$ variables whose assignment disagrees with $U_i$. We have two cases:

- $\displaystyle\sum_{i=1}^{n} t_i \le \frac{\epsilon m}{4}$: In this case, let $r \ge \epsilon m$ be the number of unsatisfied constraints of $\varphi$ and let $C_{i_1}, \ldots, C_{i_r}$ be those unsatisfied constraints. For each $1 \le l \le r$, let $Y_{l,1}$ and $Y_{l,2}$ be the two variables that appear in $C'_{i_l}$. As $\displaystyle\sum_{i=1}^{n} t_i \le \frac{\epsilon m}{4}$, at most $\frac{\epsilon m}{4}$ of the variables $Y_{1,1}, Y_{1,2}, Y_{2,1}, Y_{2,2}, \ldots, Y_{r,1}, Y_{r,2}$ disagree with the plurality-rule assignment. Therefore, there are at least $r - \frac{\epsilon m}{4} \ge \frac{3\epsilon m}{4}$ $\psi$-constraints among $C'_{i_1}, \ldots, C'_{i_r}$ which are equivalent to there $\varphi$ counterparts. Since $C_{i_1}, \ldots, C_{i_r}$ are all unsatisfied, there are at least $\frac{3\epsilon m}{4} \ge \frac{\epsilon m}{80 w_1}$ $\psi$-constraints that are not satisfied.

- $\displaystyle\sum_{i=1}^{n} t_i \le \frac{\epsilon m}{4}$: Let $T_i$ be the set of $Y_i$ variables that disagree with the plurality rule assignment (we have $|T_i| = t_i$). We have two sub-cases:
  - If $t_i \le \frac{k_i}{2}$ (remember that $k_i$ is the number of vertices in $G_i$), then according to lemma 23, by choosing an edge uniformly from $uv \in G_i$, the probability that it satisfies $u \in T_i$ and $v \notin T_i$ is at least $\frac{|T_i|}{20k_i} = \frac{t_i}{20k_i} \ge \frac{t_i}{20k_i w_1}$.
  - If $t_i > \frac{k_i}{2}$, then the complement of $T_i$ has at most $\frac{k_i}{2}$ vertices. Note that we must have $t_i \le k_i\left(1 - \frac{1}{w_1}\right)$ from the plurality rule, and the complement of $T_i$ has at least $k_i - t_i \ge \frac{k_i}{w_1} \ge \frac{t_i}{w_1}$ vertices. By applying lemma 23 to the complement of $T_i$, the probability of randomly choosing $uv \in G_i$ such that $u \in T_i$ and $v \notin T_i$, is at least $\frac{k_i - t_i}{20k_i} \ge \frac{t_i}{20k_i w_1}$.

  Therefore, in the two sub-cases, the number of edges $uv \in G_i$ satisfying $u \in T_i$ and $v \notin T_i$ is at least $\frac{t_i}{20k_i w_1} . \frac{d' k_i}{2} \ge \frac{t_i}{20 w_1}$ (here we used the fact that $d' \ge 2$ and that the total number of edges in $G_i$ is $\frac{d' k_i}{2}$ as $G_i$ is $d'$-regular). Note that if an edge $uv \in G_i$ satisfies $u \in T_i$ and $v \notin T_i$, the equality constraint corresponding to $uv$ is unsatisfied since the variable corresponding to $v$ agrees with the plurality rule, while the variable corresponding to $u$ doesn't. We conclude that the total number of unsatisfied equality constraints is at least

$$\sum_{i=1}^{n} \frac{t_i}{20 w_1} = \frac{1}{20 w_1} \sum_{i=1}^{n} t_i \le \frac{1}{20 w_1} \frac{\epsilon m}{4} = \frac{\epsilon m}{80 w_1}$$

We conclude that any assignment of the variables of $\psi$ has at least $\frac{1}{80w_1}\epsilon m$ unsatisfied constraints. And since there is a total of $d_1 m$ constraints in $\psi$, we have:

$$\mathrm{Val}(\psi) \leq 1 - \frac{\epsilon}{80w_1d_1} = 1 - \frac{1}{80w_1d_1}\Big(1 - \mathrm{Val}(\varphi)\Big) = \max\Big\{1 - \frac{1}{80w_1d_1}\Big(1 - \mathrm{Val}(\varphi)\Big), 1 - 1\Big\}.$$

This shows that $f_1$ is a $(\beta_1(w_1), 1)$-gap amplifying mapping which maps the instances of $2\mathrm{CSP}_{w_1}$ to instances of $2\mathrm{CSP}_{w_1}$ whose constraint graphs is $d_1$-regular, where $\beta_1(w_1) := \frac{1}{80w_1d_1}$. ■

**Lemma 26** *For any $w_1 \geq 2$, there exists a constant $0 < \beta(w_1) < 1$ that depends only on $w_1$, for which there exists a $(\beta(w_1), 1)$-gap amplifying mapping from $2\mathrm{CSP}_{w_1}$ to $2\mathrm{CSP}_{w_1}$ which maps any instance of $2\mathrm{CSP}_{w_1}$ to a nice instance of $2\mathrm{CSP}_{w_1}$.*

**Proof** Assume that we have an instance $\varphi$ whose constraint graph is $d_1$-regular, we need to transform it to a nice instance (i.e., we need to make its constraint graph a $d$-regular expander where at each vertex at least half of the edges are self-loops). We can achieve this by doing the following:

- Use lemma 22 to obtain a $d''(c)$-regular strong expander graph $G'$ (by a strong expander we mean a $c$-expander for a small enough $c$) spanning all the vertices of $G(\varphi)$.
- Let $\psi$ be the instance of $2\mathrm{CSP}_{w_1}$ having the same variables as $\varphi$, and which is obtained from $\varphi$ by doing the following:
  - Add all the constraints of $\varphi$.
  - For each edge $e \in G'$ connecting two variables $U_{e,1}$ and $U_{e,2}$, we add a trivial constraint on $U_{e,1}$ and $U_{e,2}$ that is always satisfied.
  - For each variable $U_i$ of $\psi$, we add $d_1 + d''(c)$ trivial constraints on $U_i$ and $U_i$ which are always satisfied.

  Therefore, $G(\psi)$ can be obtained from $G(\varphi)$ by adding $G'$ and then adding $d_1 + d''(c)$ self-loops at each vertex.

Let $f_2$ be the mapping that maps $\varphi$ to $\psi$ according to the above procedure. It is easy to see the following:

- There is only a linear increase in the size of the problem (the number of the variables remain the same, and the number of constraints is multiplied by $\frac{2(d_1+d''(c))}{d_1}$).
- $\varphi$ is satisfiable if and only if $\psi$ is satisfiable. Therefore, $f_2$ is a CL-reduction.
- $1 - \mathrm{Val}(\psi) = \frac{d_1}{2(d_1+d''(c))}(1 - \mathrm{Val}(\psi))$.
- $\psi$ is a nice instance (from the properties of expander graphs, $c$ can be chosen small enough so that by adding any $d''(c)$-regular $c$-expander graph $G'$ to any $d_1$ regular graph $G$ having the same number of vertices and then adding $d_1 + d''(c)$ self-loops at each vertex we obtain an expander graph).

This shows that $f_2$ is a $\big(\frac{d_1}{2(d_1+d''(c))}, 1\big)$-gap amplifying mapping that maps the instances of $2\mathrm{CSP}_{w_1}$ whose constraint graphs are $d_1$-regular to nice instances of $2\mathrm{CSP}_{w_1}$. By composing $f_1$ with $f_2$ (where $f_1$ is the gap amplifying mapping obtained in the previous lemma), we get a $(\beta(w_1), 1)$-gap amplifying mapping which maps the instances of $2\mathrm{CSP}_{w_1}$ to nice instances of $2\mathrm{CSP}_{w_1}$, where $\beta(w_1) := \frac{d_1}{2(d_1+d''(c))}\beta_1(w_1)$ and $\beta_1(w_1)$ is defined in the previous lemma. ■