Inapproximability in computational hardness

구재현

November 1, 2022

Joesonghabnida

소프트웨어 멤버십 과제로 만들었던 거고 번역하기 귀찮아서 한글 자료로 진행합니다.

Sorry for international audiences, if there are any.

Recap

Chapter 9에 나오는 내용을 빠르게 결과 위주로 복습해봅시다.

Recap

Chapter 9에 나오는 내용을 빠르게 결과 위주로 복습해봅시다.

Definition (PTAS)

어떠한 최소화 문제 A 에 대해 PTAS가 존재한다는 것은, 임의의 입력 x 과 상수 $\epsilon > 0$ 에 대해서, $(1+\epsilon)OPT(x)$ 이하 크기의 해를 다항 시간에 찾을 수 있음을 의미한다.

Recap

Chapter 9에 나오는 내용을 빠르게 결과 위주로 복습해봅시다.

Definition (PTAS)

어떠한 최소화 문제 A 에 대해 PTAS가 존재한다는 것은, 임의의 입력 x 과 상수 $\epsilon > 0$ 에 대해서, $(1+\epsilon)OPT(x)$ 이하 크기의 해를 다항 시간에 찾을 수 있음을 의미한다.

Definition (복잡도 클래스)

1. PTAS 는 PTAS가 존재하는 문제들의 집합이다.
2. 최소화 문제 $A \in$ APX 라 함은, 어떠한 $c \geq 1$ 이 존재하여 $c \times OPT(x)$ 이하의 해를 다항 시간에 찾을 수 있음을 뜻한다.
3. 최소화 문제 $A \in$ Log-APX 라 함은, 어떠한 $c \geq 1$ 이 존재하여 $c \log x \times OPT(x)$ 이하의 해를 다항 시간에 찾을 수 있음을 뜻한다.
4. 최소화 문제 $A \in$ Poly-APX 라 함은, 어떠한 다항식 p 가 존재하여 $p(x) \times OPT(x)$ 이하의 해를 다항 시간에 찾을 수 있음을 뜻한다.

Recap

$P \neq NP$ 를 가정하면 다음과 같은 사실이 참입니다.

1. TSP $\notin$ Poly-APX
2. CLIQUE $\in$ Poly-APX $-$ Log-APX
3. SET-COVER $\in$ Log-APX $-$ APX
4. MAX-3SAT $\in$ APX $-$ PTAS

즉, 네 개의 복잡도 클래스는 모두 $\subsetneq$ 관계입니다.

몰라도 되지만 마지막 MAX-3SAT 는 기억하셔야 합니다.

APX-Hard, APX-Complete

Recap: Cook-Levin Theorem

Theorem (Cook-Levin)

3-SAT 문제는 NP이며, 모든 다른 NP 문제를 3-SAT으로 다항 시간에 reduction할 수 있다.

Recap: Cook-Levin Theorem

Theorem (Cook-Levin)

3-SAT 문제는 NP이며, 모든 다른 NP 문제를 3-SAT으로 다항 시간에 reduction할 수 있다.

Cook-Levin Theorem의 프레임워크 안에서 결과를 만드는 방법은 Polynomial-time reduction 입니다.

Recap: Cook-Levin Theorem

Theorem (Cook-Levin)

3-SAT 문제는 NP이며, 모든 다른 NP 문제를 3-SAT으로 다항 시간에 reduction할 수 있다.

Cook-Levin Theorem의 프레임워크 안에서 결과를 만드는 방법은 Polynomial-time reduction 입니다.

Approximation에서 이 개념들을 그대로 적용하기에는 번거로움이 있습니다. **Approximation factor** 라는 추가적인 인자를 고려해야 하기 때문입니다.

이 부분을 잘 녹여낸 형태의 Reduction에 대해서 생각해 봅시다.

Definition: Approximation Preserving Reduction

A, B 를 두 개의 최적화 문제라고 하자. A 에서 B 로 가는 *approximation preserving reduction (APR)* 은 다음 성질을 만족하는, 다항 시간에 계산 가능한 함수 (f, g) 의 쌍이다:

Definition: Approximation Preserving Reduction

A, B 를 두 개의 최적화 문제라고 하자. A 에서 B 로 가는 *approximation preserving reduction (APR)* 은 다음 성질을 만족하는, 다항 시간에 계산 가능한 함수 (f, g) 의 쌍이다:

1. A 의 입력 x 에 대응되는 B 의 입력 $x' = f(x)$ 이 항상 존재한다.
2. 문제 B 에 대한 입력 x' 의 해 y 가 주어질 때, $y' = g(x, y)$ 는 문제 A에 대한 입력 x 의 해이다.
3. y'는 y 와 **비슷하게 좋은 해** 이다. 이 *비슷한 정도* 가 어떤지에 따라서 세부적인 Reduction이 결정된다.

Definition: Approximation Preserving Reduction

A, B 를 두 개의 최적화 문제라고 하자. A 에서 B 로 가는 *approximation preserving reduction (APR)* 은 다음 성질을 만족하는, 다항 시간에 계산 가능한 함수 (f, g) 의 쌍이다:

1. A 의 입력 x 에 대응되는 B 의 입력 $x' = f(x)$ 이 항상 존재한다.
2. 문제 B 에 대한 입력 x' 의 해 y 가 주어질 때, $y' = g(x, y)$ 는 문제 A에 대한 입력 x 의 해이다.
3. y'는 y 와 **비슷하게 좋은 해** 이다. 이 *비슷한 정도* 가 어떤지에 따라서 세부적인 Reduction이 결정된다.
4. 다음과 같은 예시가 있다:
   - 4.1 **PTAS reduction** 은, 임의의 $\epsilon > 0$ 에 대해서, 무한으로 가는 증가함수 $\delta(\epsilon) > 0$이 존재하여, 만약 y' 이 B 에 대한 $(1 + \delta(\epsilon))$-approximation 이라면, y 가 A 에 대한 $(1 + \epsilon)$-approximation 이어야 한다.
   - 4.2 **Strict reduction** 은, $\delta(\epsilon) = \epsilon$ 인 PTAS reduction 이다.
   - 4.3 **APX reduction** 은, $\delta(\epsilon) = O(\epsilon)$ 인 PTAS reduction 이다.

Definition: Approximation Preserving Reduction

1. A 의 입력 x 에 대응되는 B 의 입력 $x' = f(x)$ 이 항상 존재한다.
2. 문제 B 에 대한 입력 x' 의 해 y 가 주어질 때, $y' = g(x, y)$ 는 문제 A에 대한 입력 x 의 해이다.
3. y'는 y 와 **비슷하게 좋은 해** 이다. 이 *비슷한 정도* 가 어떤지에 따라서 세부적인 Reduction이 결정된다.
4. 다음과 같은 예시가 있다:
   - 4.1 **PTAS reduction** 은, 임의의 $\epsilon > 0$ 에 대해서, 무한으로 가는 증가함수 $\delta(\epsilon) > 0$이 존재하여, 만약 y' 이 B 에 대한 $(1 + \delta(\epsilon))$-approximation 이라면, y 가 A 에 대한 $(1 + \epsilon)$-approximation 이어야 한다.
   - 4.2 **Strict reduction** 은, $\delta(\epsilon) = \epsilon$ 인 PTAS reduction 이다.
   - 4.3 **APX reduction** 은, $\delta(\epsilon) = O(\epsilon)$ 인 PTAS reduction 이다.

APR 자체는 수학적 정의가 아니지만, *비슷하게 좋다* 의 기준을 설정함으로서 수학적 정의로 완성됩니다.

Definition: Approximation Preserving Reduction

1. A 의 입력 x 에 대응되는 B 의 입력 $x' = f(x)$ 이 항상 존재한다.
2. 문제 B 에 대한 입력 x' 의 해 y 가 주어질 때, $y' = g(x, y)$ 는 문제 A에 대한 입력 x 의 해이다.
3. y'는 y 와 **비슷하게 좋은 해** 이다. 이 *비슷한 정도* 가 어떤지에 따라서 세부적인 Reduction이 결정된다.
4. 다음과 같은 예시가 있다:
   - 4.1 **PTAS reduction** 은, 임의의 $\epsilon > 0$ 에 대해서, 무한으로 가는 증가함수 $\delta(\epsilon) > 0$이 존재하여, 만약 y' 이 B 에 대한 $(1 + \delta(\epsilon))$-approximation 이라면, y 가 A 에 대한 $(1 + \epsilon)$-approximation 이어야 한다.
   - 4.2 **Strict reduction** 은, $\delta(\epsilon) = \epsilon$ 인 PTAS reduction 이다.
   - 4.3 **APX reduction** 은, $\delta(\epsilon) = O(\epsilon)$ 인 PTAS reduction 이다.

APR 자체는 수학적 정의가 아니지만, *비슷하게 좋다* 의 기준을 설정함으로서 수학적 정의로 완성됩니다.

PTAS-reduction은 PTAS membership을 보존하고, APX-reduction은 APX membership (및 PTAS membership) 을 보존합니다. 증명은 쉽습니다 (숙제에 있음).

APX-hard, APX-complete

Definition (APX-hard)

어떠한 문제 B가 APX-hard 라는 것은, MAX-3SAT에서 B 로 가는 APX reduction이 존재함을 뜻한다.

Definition (APX-complete)

어떠한 문제 B가 APX-complete 라는 것은, B 가 APX-hard이며 $B \in$ APX 임을 뜻한다.

APX-hard, APX-complete

Definition (APX-hard)

어떠한 문제 B가 APX-hard 라는 것은, MAX-3SAT에서 B 로 가는 APX reduction이 존재함을 뜻한다.

Definition (APX-complete)

어떠한 문제 B가 APX-complete 라는 것은, B 가 APX-hard이며 $B \in$ APX 임을 뜻한다.

Theorem (Cook-Levin. 1971)

3-SAT 문제는 NP이며, 모든 다른 NP 문제를 3-SAT으로 Polynomial-time reduction 할 수 있다.

Theorem (Papadimitriou-Yannakakis. 1991)

MAX-3SAT 문제는 APX이며, 모든 다른 APX 문제를 MAX-3SAT 문제로 APX-reduction할 수 있다.

APX-hard, APX-complete

Definition (APX-hard)

어떠한 문제 B가 APX-hard 라는 것은, MAX-3SAT에서 B 로 가는 APX reduction이 존재함을 뜻한다.

Definition (APX-complete)

어떠한 문제 B가 APX-complete 라는 것은, B 가 APX-hard이며 $B \in$ APX 임을 뜻한다.

Corollary

$B \in$ APX-Complete 일 경우

1. $B \in$ APX.
2. $B \in$ PTAS 일 경우 P $=$ NP.

L-reduction

PTAS reduction과는 조금 다른 형태의 L-reduction을 소개합니다. PTAS reduction보다 사용하기 편하고 조금 더 범용적이라 여기서부터는 L-reduction만을 사용합니다.

L-reduction

PTAS reduction과는 조금 다른 형태의 L-reduction을 소개합니다. PTAS reduction보다 사용하기 편하고 조금 더 범용적이라 여기서부터는 L-reduction만을 사용합니다.

Definition (L-reduction)

A 에서 B 로의 L-reduction $A \leq_L B$ 는 다음 조건을 만족하는 APR 이다:

1. $OPT_B(x') = O(OPT_A(x))$
2. $|cost_A(y) - OPT_A(x)| = O(|cost_B(y') - OPT_B(x')|)$

L-reduction

PTAS reduction과는 조금 다른 형태의 L-reduction을 소개합니다. PTAS reduction보다 사용하기 편하고 조금 더 범용적이라 여기서부터는 L-reduction만을 사용합니다.

Definition (L-reduction)

A 에서 B 로의 L-reduction $A \leq_L B$ 는 다음 조건을 만족하는 APR 이다:

1. $OPT_B(x') = O(OPT_A(x))$
2. $|cost_A(y) - OPT_A(x)| = O(|cost_B(y') - OPT_B(x')|)$

Fact

$A \leq_L B$ 일 경우 A 에서 B 로 가는 APX reduction이 존재한다.

Fact

$A \leq_L B, B \leq_L C$ 일 경우 $A \leq_L C$ 이다.

두 Fact의 증명은 과제에서 직접 해 보시게 될 겁니다.

Max 3SAT, it's variants, graph counterparts

MAX 3SAT Variants

MAX-3SAT 문제에 몇 가지 제약을 더 건, *쉬운* 변형들을 살펴보고, 이 문제들 역시 모두 APX-complete 임을 보입니다.

Definition (MAX-3SAT)

모든 clause가 ≤ 3 개의 literal을 가진 formula가 주어질 때, 만족 가능한 clause 수의 최댓값을 계산하여라.

MAX 3SAT Variants

Definition (MAX-3SAT-Ea)

모든 clause가 ≤ 3 개의 literal을 가지고, 각 변수가 최대 $\leq a$ 번 등장하는 formula가 주어질 때, 만족 가능한 clause 수의 최댓값을 계산하여라.

MAX 3SAT Variants

Definition (MAX-3SAT-Ea)

모든 clause가 ≤ 3 개의 literal을 가지고, 각 변수가 최대 $\leq a$ 번 등장하는 formula가 주어질 때, 만족 가능한 clause 수의 최댓값을 계산하여라.

Definition (MAX-2SAT)

모든 clause가 ≤ 2 개의 literal을 가진 formula가 주어질 때, 만족 가능한 clause 수의 최댓값을 계산하여라.

MAX 3SAT Variants

Definition (MAX-3SAT-Ea)

모든 clause가 ≤ 3 개의 literal을 가지고, 각 변수가 최대 $\leq a$ 번 등장하는 formula가 주어질 때, 만족 가능한 clause 수의 최댓값을 계산하여라.

Definition (MAX-2SAT)

모든 clause가 ≤ 2 개의 literal을 가진 formula가 주어질 때, 만족 가능한 clause 수의 최댓값을 계산하여라.

Definition (MAX-NAE-3SAT)

모든 clause가 ≤ 3 개의 literal을 가진 formula가 주어질 때, 어떠한 clause도 모든 literal이 참이 되지 않게 하면서, 만족 가능한 clause 수의 최댓값을 계산하여라.

MAX 3SAT Variants

오늘의 Main Theorem입니다.

Theorem

1. *MAX-3SAT* $\leq_L$ *MAX-3SAT-E7*
2. *MAX-3SAT-E7* $\leq_L$ *MAX-3SAT-E3*
3. *MAX-3SAT-E3* $\leq_L$ *MAX-3SAT*

MAX 3SAT Variants

오늘의 Main Theorem입니다.

Theorem

1. *MAX-3SAT* $\leq_L$ *MAX-3SAT-E7*
2. *MAX-3SAT-E7* $\leq_L$ *MAX-3SAT-E3*
3. *MAX-3SAT-E3* $\leq_L$ *MAX-3SAT*

Corollary

MAX-3SAT $\leq_L$ MAX-3SAT-E3 (Transivity fact에서 유도)

Corollary

MAX-3SAT-E3 은 APX-complete이다. (APX-Complete 정의 및 MAX-3SAT $\in$ APX.)

MAX 3SAT Variants

만만하지 않습니다. 일단 잘 알려진 쉬운 Lemma부터 소개합니다.

Lemma

C 개의 clause를 가진 3SAT formula가 주어질 때, $O(C)$ 개 이상의 clause를 만족시키는 배정이 존재한다.

MAX 3SAT Variants

만만하지 않습니다. 일단 잘 알려진 쉬운 Lemma부터 소개합니다.

Lemma

C 개의 clause를 가진 3SAT formula가 주어질 때, $O(C)$ 개 이상의 clause를 만족시키는 배정이 존재한다.

Proof

각 clause에서 하나의 변수만 남겨두자. 모든 변수에 대해서 랜덤하게 배정했을 때, 각 clause가 참으로 계산될 확률은 최소 0.5 이다.

이는 clause에서 하나의 변수만 남겼을 때의 결과이니 하나 이상일 경우에 확률이 감소하지 않는다.

기댓값의 선형성에 의해, 참이 되는 clause의 기댓값은 $0.5C$ 이다. 고로 $0.5C$ 개 이상의 clause를 만족시키는 배정이 존재한다.

MAX 3SAT Variants

만만하지 않습니다. 일단 잘 알려진 쉬운 Lemma부터 소개합니다.

Lemma

C 개의 clause를 가진 3SAT formula가 주어질 때, $O(C)$ 개 이상의 clause를 만족시키는 배정이 존재한다.

Proof

각 clause에서 하나의 변수만 남겨두자. 모든 변수에 대해서 랜덤하게 배정했을 때, 각 clause가 참으로 계산될 확률은 최소 0.5 이다.

이는 clause에서 하나의 변수만 남겼을 때의 결과이니 하나 이상일 경우에 확률이 감소하지 않는다.

기댓값의 선형성에 의해, 참이 되는 clause의 기댓값은 $0.5C$ 이다. 고로 $0.5C$ 개 이상의 clause를 만족시키는 배정이 존재한다.

MAX 3SAT Variants

이제 잘 알려지지 않고 어려운 Lemma를 소개합니다.

Definition (Expander Graph)

$d \in \mathbb{N}$ 에 대해서, d-expander graph $G = (V, E)$ 는 모든 정점의 차수가 d 이며, 모든 정점의 이분할 $V = V_1 \cup V_2$ 에 대해 V_1 과 V_2 사이를 잇는 간선의 수가 $min(|V_1|, |V_2|)$ 이상이다.

MAX 3SAT Variants

이제 잘 알려지지 않고 어려운 Lemma를 소개합니다.

Definition (Expander Graph)

$d \in \mathbb{N}$ 에 대해서, d-expander graph $G = (V, E)$ 는 모든 정점의 차수가 d 이며, 모든 정점의 이분할 $V = V_1 \cup V_2$ 에 대해 V_1 과 V_2 사이를 잇는 간선의 수가 $min(|V_1|, |V_2|)$ 이상이다.

Lemma

모든 $k \equiv 0 \pmod 2$ 에 대해, k 개의 정점을 가진 3-expander 가 존재한다.

MAX 3SAT Variants

이제 잘 알려지지 않고 어려운 Lemma를 소개합니다.

Definition (Expander Graph)

$d \in \mathbb{N}$ 에 대해서, d-expander graph $G = (V, E)$ 는 모든 정점의 차수가 d 이며, 모든 정점의 이분할 $V = V_1 \cup V_2$ 에 대해 V_1 과 V_2 사이를 잇는 간선의 수가 $min(|V_1|, |V_2|)$ 이상이다.

Lemma

모든 $k \equiv 0 \pmod 2$ 에 대해, k 개의 정점을 가진 3-expander 가 존재한다.

책에서 Exercise로 나와있던데 참인지 잘 모르겠습니다. 원본 논문에서는 3이 아니라 *어떤 유한한 상수* 로 뒀습니다. 14-expander 에 대해서는 괜찮은 결과가 있었고 몇몇 교과서에서는 이 결과를 활용했습니다.

어쨌든 유한한 상수라고만 쳐도 증명이 달라지지 않으니, 그냥 그렇다 치고 넘어갑시다.

MAX 3SAT Variants

Proof of Main Theorem. Part 1:

WLOG 모든 변수가 짝수번 등장한다고 가정하자. 8 번 이상 등장하는 모든 변수 x에 대해 다음과 같이 변환한다:

1. x 가 k 번 등장한다고 하면 $z_1, \ldots, z_k$ 라는 새로운 변수를 만들고 모든 x 의 occurence를 $z_1, \ldots, z_k$ 로 대체한다.
2. G 를 k 개의 정점을 가진 3-expander graph 라고 하자 (Lemma). 모든 간선 $\{i, j\}$ 에 대해 clause $(z_i \to z_j), (z_j \to z_i)$ 를 추가한다.

그래프의 차수가 정확히 3 이기 때문에, 각 z_i 가 정확히 7번 등장한다.

이로서 MAX-3SAT-E7 에 대응되는 인스턴스 x' 을 만들었다. 이 인스턴스의 해 y'에서 원래 인스턴스의 해 y 를 이끌어내자.

우리가 원하는 상황은 $z_1, \ldots, z_k$ 이 y' 에서 모두 같은 값으로 배정되어 있는 것인데, 실제로는 그렇지 않을 수 있다.

MAX 3SAT Variants

인스턴스 x' 의 최적해 중 z_i 에 대한 배정이 모두 동일한 것이 존재한다.

Z_{TRUE} 를 z_i 에 참이 배정된 집합, Z_{FALSE} 를 z_i 에 거짓이 배정된 집합이라고 하고, WLOG $|Z_{TRUE}| \geq |Z_{FALSE}| \geq 1$ 이라 하자.

Z_{FALSE} 에서 Z_{TRUE} 로 가는 간선은 최소 $|Z_{FALSE}|$ 개 존재하며, 고로 현재 Expander graph에서 $|Z_{FALSE}|$ 개의 clause가 만족되지 않고 있다. 이들을 모두 True로 바꾸면 해당 개수만큼의 clause를 새로 만족시킨다.

한편, 각각의 z_i 는 Expander graph가 아닌 곳에서 단 한번 등장했기 때문에 최대 $|Z_{FALSE}|$ 개의 clause가 새롭게 만족되지 않는다.

이상의 Reduction이 L-reduction임을 증명하면 된다.

MAX 3SAT Variants

1. $OPT_B(x') = O(OPT_A(x))$: 첫 Lemma의 결과로 유도할 수 있다. x 가 C 개의 Clause를 가진다면, $O(OPT_A(x)) = O(C)$ 이고, x' 역시 $O(C)$ 개의 Clause를 가지니, $OPT_B(x') = O(C)$ 이다.
2. $OPT_A(x) + 3\sum_i k_i = OPT_B(x')$: $z_i = x$ 로 동일하게 배정할 경우 새로 추가한 모든 Clause가 만족된다. $OPT_B(x')$ 의 최적해는 모든 z_i 에 대한 배정이 동일하다 가정할 수 있다.
3. $cost_A(y) + 3\sum_i k_i \geq cost_B(y')$: y' 을 위에서 설명한 방법대로 transform할 경우 항상 비용이 좋아진다. 그 이후에는 z_i 에 대한 배정이 동일하니 그대로 A 의 최적해로 사용하면 된다.

$$OPT_A(x) + 3\sum_i k_i = OPT_B(x')$$

$$0 \geq cost_A(y) - OPT_A(x) \geq cost_B(y') - OPT_B(x')$$

$$|cost_A(y) - OPT_A(x)| = O(|cost_B(y') - OPT_B(x')|) \blacksquare$$

MAX 3SAT Variants

1. $OPT_B(x') = O(OPT_A(x))$: 첫 Lemma의 결과로 유도할 수 있다. x가 C 개의 Clause를 가진다면, $O(OPT_A(x)) = O(C)$ 이고, x' 역시 $O(C)$ 개의 Clause를 가지니, $OPT_B(x') = O(C)$ 이다.
2. $OPT_A(x) + 3\sum_i k_i = OPT_B(x')$: $z_i = x$ 로 동일하게 배정할 경우 새로 추가한 모든 Clause가 만족된다. $OPT_B(x')$ 의 최적해는 모든 z_i 에 대한 배정이 동일하다 가정할 수 있다.
3. $cost_A(y) + 3\sum_i k_i \geq cost_B(y')$: y' 을 위에서 설명한 방법대로 transform할 경우 항상 비용이 좋아진다. 그 이후에는 z_i 에 대한 배정이 동일하니 그대로 A 의 최적해로 사용하면 된다.

$$OPT_A(x) + 3\sum_i k_i = OPT_B(x')$$

$$0 \geq cost_A(y) - OPT_A(x) \geq cost_B(y') - OPT_B(x')$$

$$|cost_A(y) - OPT_A(x)| = O(|cost_B(y') - OPT_B(x')|) \blacksquare$$

Part 2는 아마 저희가 배운 내용 안에서 증명하지 못할 겁니다. Part 3은 자명합니다.

Graph Counterparts

이제 MAX-3SAT-E3 에 대한 분석을 끝냈으니, 이 사실을 사용하여 몇 가지 잘 알려진 그래프 문제에 대한 Hardness를 증명합니다.

Definition (INDEPENDENT SET-a)

모든 정점의 차수가 $\leq a$ 인 그래프가 주어질 때 최대 독립 집합의 크기를 반환한다.

Definition (VERTEX COV-a)

모든 정점의 차수가 $\leq a$ 인 그래프가 주어질 때 최대 독립 집합의 크기를 반환한다.

Definition (MAXCUT)

그래프가 주어질 때, $V = V_1 \cup V_2$ 인 이분할 (V_1, V_2) 중 V_1 과 V_2 사이를 잇는 최대 간선의 개수를 반환한다.

Graph Counterparts

Theorem

다음과 같은 사실이 모두 참이다. 편의상 $P \neq NP$ 를 가정한다.

1. *MAX-3SAT-E3* $\leq_L$ *INDEPENDENT SET-*4
2. *모든* $\Delta \geq 4$ *에 대해 INDEPENDENT SET-*Δ *는 APX-Complete 이다.*
3. *INDEPENDENT SET-*$4 \leq_L$ *MAX-2SAT*
4. *MAX-2SAT* $\leq_L$ *MAX-NAE-3SAT*
5. *MAX-NAE-3SAT* $\leq_L$ *MAX-CUT*
6. *MAX-CUT* $\in APX - PTAS$. *이는 또한 MAX-CUT 이 APX-Complete 임을 증명한다.*

Graph Counterparts

Theorem

다음과 같은 사실이 모두 참이다. 편의상 $P \neq NP$ 를 가정한다.

1. *MAX-3SAT-E3 $\leq_L$ INDEPENDENT SET-4*
2. *모든 $\Delta \geq 4$ 에 대해 INDEPENDENT SET-Δ 는 APX-Complete 이다.*
3. *INDEPENDENT SET-$4 \leq_L$ MAX-2SAT*
4. *MAX-2SAT $\leq_L$ MAX-NAE-3SAT*
5. *MAX-NAE-3SAT $\leq_L$ MAX-CUT*
6. *MAX-CUT $\in APX - PTAS$. 이는 또한 MAX-CUT 이 APX-Complete 임을 증명한다.*

이걸 다 증명하는 건 너무 시간이 많이 걸리기도 하고, 사실 앞선 Theorem만큼 그렇게 어렵지 않아서 좋은 연습문제라고 생각합니다.

발표에서는 제일 어려운 Part 5, 6 정도만 증명합니다.

Graph Counterparts

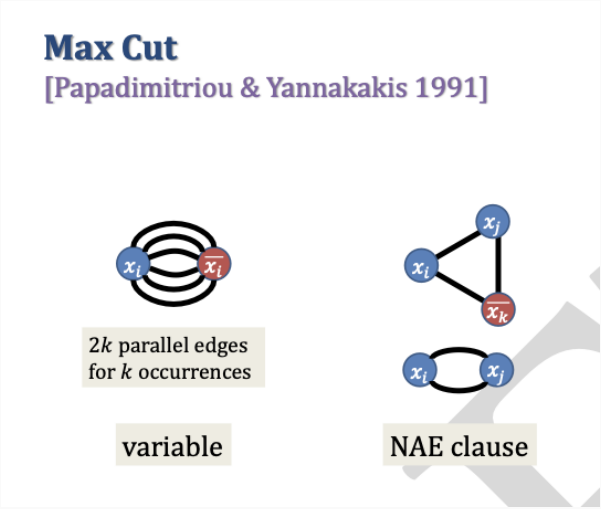
Proof of Theorem. Part 5: MAX-NAE-3SAT의 각 Clause에 중복된 원소가 없고, 1-clause가 없음을 가정하자. (이렇게 하여도 Part 4의 Specification과 모순이 없다.)

수식 x 가 주어졌을 때 다음과 같은 그래프 x' 을 만든다. x'에는 중복 간선이 있을 수 있음을 유념하라.

1. 모든 변수 x 에 대해, $x, \neg x$ 라는 정점을 만들고, $k_i = 2\times$($x, \neg x$ 의 등장 횟수 합)만큼 두 정점간에 중복 간선을 추가해준다.
2. clause $L_1 \vee L_2 \vee L_3$ 에 대해서, 모든 L_i 와 L_j 간에 간선을 하나 잇는다.
3. clause $L_1 \vee L_2$ 에 대해서, L_1, L_2 간에 간선을 두개 잇는다.

이 인스턴스에 대한 정점 이분할 $y' = (V_1, V_2)$ 가 주어질 때 이를 x 에 대한 assignment y 로 변환한다.

Graph Counterparts



만약 $x, \neg x$ 가 같은 사이드에 있지 않다면 assignment y 가 비교적 명확히 드러난다.

그렇지 않을 경우가 문제인데, Clause에 의해서 생기는 간선은 각 등장마다 정확히 2개이기 때문에, 중복 간선의 개수보다 항상 작거나 같다. 고로 정점 $x, \neg x$ 가 같은 사이드에 있다면, 그 중 임의로 하나를 다른 사이드로 배정할 경우 답이 나빠지지 않는다.

이제 $x \in V_1$ 인 변수들에 대해서 참을 배정하면 된다.

Graph Counterparts

1. 각 Clause에 1-clause가 없고 중복된 원소가 없기 때문에, clause 하나만을 만족시키는 assignment는 항상 존재한다. 고로 MAX-NAE-3SAT 역시 Randomization을 통해서 $O(|C|)$ 크기의 해를 찾을 수 있다. $\sum k_i = O(|C|)$ 이니 $OPT_B(x') = O(OPT_A(x))$.
2. NAE Clause 하나는 Max Cut 상의 2개의 간선에 기여한다. 또한 $x, \neg x$ 사이의 중복 간선은 모두 Max Cut에 들어가게 된다. 고로 $OPT_A(x) \times 2 + \sum k_i \leq OPT_B(x')$
3. Max Cut의 최적해는 중복 간선을 모두 사용하며, 각 NAE Clause에 대해서 2개의 간선을 사용하거나 사용하지 않는다. 우리의 변환 과정에 의해, 2개의 간선을 사용한 NAE Clause는 모두 만족되며, 그렇지 않은 NAE Clause는 만족되지 않는다. 고로 $OPT_A(x) \times 2 + \sum k_i \geq OPT_B(x')$

이 사실을 토대로 전개 시 L-reduction임을 보일 수 있다.

Graph Counterparts

Proof of Theorem. Part 6: MaxCut 문제는 APX-hard이기 때문에 $P \neq NP$ 가정에 의해 PTAS에 속하지 않는다. MaxCut 문제는 0.878-approximation이 존재하는데, 0.5 approximation이 더 간단하기 때문에 여기서는 이를 소개한다.

각 정점이 속할 bipartition을 단순히 coin flip으로 랜덤하게 결정하면, 최적해의 간선이 해당 bipartition의 비용에 기여할 확률이 0.5 이다. 기댓값의 선형성에 의해, 이러한 랜덤 시행을 여러번 해보면 $0.5OPT$ 이상의 답을 얻을 수 있다.

고로 MAX-CUT $\in$ APX 이다.

Bonus Remarks

APX-Intermediate

단순 그래프가 주어질 때, **간선**에 색을 칠해서, 모든 정점에 대해 인접한 간선들이 전부 다른 색을 가지게 하는 **간선 채색** (Edge Coloring) 문제를 생각해 봅시다. 목표는 사용한 서로 다른 색의 개수 최소화입니다.

최대 차수가 Δ 일 때 최소 Δ 개의 색이 필요합니다.

APX-Intermediate

단순 그래프가 주어질 때, **간선**에 색을 칠해서, 모든 정점에 대해 인접한 간선들이 전부 다른 색을 가지게 하는 **간선 채색** (Edge Coloring) 문제를 생각해 봅시다. 목표는 사용한 서로 다른 색의 개수 최소화입니다.

최대 차수가 Δ 일 때 최소 Δ 개의 색이 필요합니다.

Theorem (Vizing, Misra-Gries)

그래프의 최대 차수가 Δ 일 때 $\Delta + 1$ 개의 색으로 간선 채색을 항상 할 수 있으며, 이를 다항 시간에 찾는 알고리즘이 존재한다. 한편, Δ 개의 색을 사용한 간선 채색의 존재성을 판별하는 것은 NP-hard이다.

고로 이 문제는 APX $-$ PTAS 에 속합니다. 하지만 APX-Complete에 속하는지가 알려져 있지 않습니다.

APX-Intermediate

단순 그래프가 주어질 때, **간선**에 색을 칠해서, 모든 정점에 대해 인접한 간선들이 전부 다른 색을 가지게 하는 **간선 채색** (Edge Coloring) 문제를 생각해 봅시다. 목표는 사용한 서로 다른 색의 개수 최소화입니다.

최대 차수가 Δ 일 때 최소 Δ 개의 색이 필요합니다.

Theorem (Vizing, Misra-Gries)

그래프의 최대 차수가 Δ 일 때 $\Delta + 1$ 개의 색으로 간선 채색을 항상 할 수 있으며, 이를 다항 시간에 찾는 알고리즘이 존재한다. 한편, Δ 개의 색을 사용한 간선 채색의 존재성을 판별하는 것은 NP-hard이다.

고로 이 문제는 APX $-$ PTAS 에 속합니다. 하지만 APX-Complete에 속하는지가 알려져 있지 않습니다.

이러한 문제를 **APX-Intermediate** 라고 합니다. 소인수 분해, 이산 로그 등이 속한 **NP-Intermediate** 에 대응됩니다. Bin Packing 문제가 APX-Intermediate에 속합니다.

Other Approximation Hardness Classes

Log-APX-Complete는, Dominating Set에서 L-reduction 되면서 Log-APX에 속하는 문제들입니다. Set Cover, Node-Weighted Steiner Tree 등이 이 부류에 속한다.

Poly-APX-Complete는, Log-APX와 유사하게 정의됩니다. 최대 클리크와 최대 독립집합이 이 분류에 속합니다. 앞서 보인 최대 독립 집합은 Bounded degree를 가정하였기 때문에 이 경우와는 상황이 다릅니다.

EXPTIME-APX-Complete도 존재합니다. TSP가 이 경우에 속하는데, 간선의 가중치가 입력 크기에 지수적일 수 있기 때문입니다.