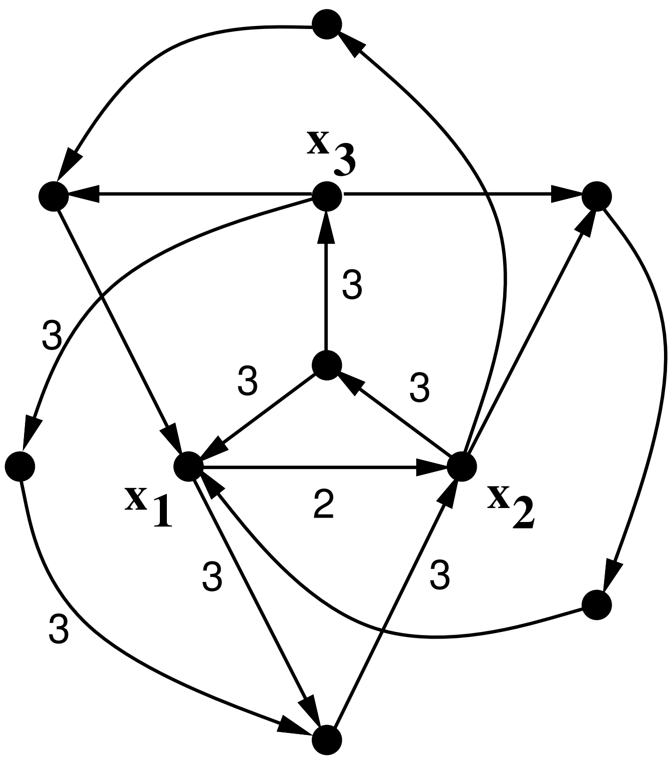
UNIVERSITÀ DEGLI STUDI DI ROMA "LA SAPIENZA"
DOTTORATO DI RICERCA IN INFORMATICA, IX CICLO

Riduzioni e Non-approssimabilità

Tesi di Dottorato

LUCA TREVISAN
Dipartimento di Scienze dell'Informazione
Via Salaria 113, 00198 Roma
E-mail: `trevisan@dsi.uniroma1.it`
URL: `http://dsi.uniroma1.it/~trevisan`



Indice

Prefazione

La presente tesi ha come argomento la complessità delle versioni approssimate di problemi di ottimizzazione. A partire dal 1991, tale area di ricerca ha compiuto straordinari progressi, soprattutto grazie ai contributi provenienti dal campo della *verifica probabilistica di dimostrazioni.* Il Capitolo 1 fornisce una storia aggiornata del campo e una presentazione sintetica dei nostri risultati. Il Capitolo 2 è dedicato a richiami di nozioni e di risultati che saranno usati nel seguito. I risultati originali esposti nel Capitolo 3 riguardano lo sviluppo di un contesto teorico per la classificazione delle proprietà di approssimazione dei problemi di ottimizzazione. Due nozioni classiche della teoria della complessità giocano un ruolo fondamentale: riduzioni e completezza. Il Capitolo 4 è dedicato a nuove tecniche di riduzione che usiamo per migliorare precedenti risultati di non-approssimabilità. Proponiamo alcune questioni aperte e congetture nel Capitolo 5.

Ringraziamenti

Sono riconoscente al mio tutore Pierluigi Crescenzi per essere stato tutore, coautore, collega and amico, per avermi insegnato tutto quel che so su come fare ricerca, e per avermi sempre indicato problemi interessanti e stimolanti. Grazie a Daniel Bovet per avermi fatto conoscere la teoria della complessità e per avermi presentato a Pierluigi.

Durante una visita breve ma eccitante al centro di ricerca *T.J. Watson* dell'IBM, Madhu Sudan mi ha insegnato molte cose sulle dimostrazioni verificabili probabilisticamente, sulla programmazione semidefinita e sul mangiar fuori a Manhattan.

Sono anche riconoscente a José Rolim, che mi ha ospitato durante una visita all'Università di Ginevra, e a José Balcázar che mi ha ospitato durante una visita all'Università Politecnica della Catalogna.

I risultati esposti in questa tesi sono frutto di lavoro comune con Pierluigi, Madhu, Andrea Clementi, Viggo Kann, Riccardo Silvestri, Greg Sorkin and David Williamson [38, 34, 30, 37, 107, 39]. In aggiunta, due teoremi sono presi da [106].

Ringrazio Giorgio Ausiello, per aver partecipato con Pierluigi e Riccardo al mio comitato tesi. Mihir Bellare e Alessandro Panconesi sono stati i revisori esterni di questa tesi.

Il mio lavoro ha benificiato di numerose costruttive conversazioni con molte altre persone, tra le quali Paola Alimonti, Marco Cesati, Richard Chang, Josep Díaz, Ray Greenlaw, Sanjeev Khanna, Hoong Chuin Lau, Wolfgang Merkle, Chiara Petrioli, e Fatos Xhafa.

I miei colleghi dottorandi e post-doc al Dipartimento di Scienze dell'Informazione e al Dipartimento di Informatica e Sistemistica dell'Università di Roma "La Sapienza" meritano uno speciale ringraziamento per avermi fatto passare gli ultimi tre anni in un ambiente di lavoro piacevole e amichevole. Mi devo scusare con loro per la mia pigrizia.

In questi anni molti altri amici mi hanno aiutato a crescere come persona e come ricercatore, e spiace constatare quanto poco successo abbiano avuto i loro sforzi. Non cercherò di elencare i loro nomi, perché qualunque elenco del genere sarebbe incompleto, ma li ringrazio tutti di cuore e li prego di non desistere.

I miei genitori sono certamente le persone che hanno più contribuito a questa tesi. Sono riconoscente verso di loro per il loro amore e la loro cura nei miei confronti.

Capitolo 1

Introduzione

Once more, we have decreased the number of open questions in the field – without, alas, increasing much the number of answers!
(C.H. Papadimitriou e M. Yannakakis [94])

In un problema di *ottimizzazione combinatoria*, è data una *istanza* alla quale è associato un insieme di soluzioni *ammissibili*. Una funzione di *misura* (o funzione *obiettivo*) appropriatamente definita attribuisce un *costo* (un intero positivo) a ciascuna di tali soluzioni, e l'obiettivo è di trovare una soluzione ammissibile di costo ottimo (massimo o minimo). Per esempio, in MIN TSP (Minimum Traveling Salesman Problem, problema del commesso viaggiatore) è dato un insieme di "città" $\{c_1, \ldots, c_n\}$ e, per ogni coppia di città (c_i, c_j), la loro "distanza" $d_{i,j}$. L'obiettivo è trovare un ciclo di lunghezza minima che tocchi ogni città una e una sola volta.

Centinaia di problemi naturali in Informatica, Ricerca Operativa, e Ingegneria (ma anche Fisica Teorica, Matematica Discreta, Biologia Molecolare, etc.) sono formulati in questo modo. Il fatto che lo spazio di ricerca sia finito suggerisce un semplice algoritmo per risolvere tali problemi: esaminare tutte le soluzioni ammissibili e restituire la migliore. Il problema è che la dimensione dello spazio di ricerca è tipicamente esponenziale nella dimensione dell'input: in MIN TSP ci sono $n!$ soluzioni ammissibili per un'istanza con n città. Per alcuni problemi, tuttavia, è possibile trovare algoritmi non banali che trovino efficientemente[1] una soluzione ottima.

Alcuni di tali algoritmi hanno rappresentato alcuni dei migliori successi iniziali dell'algoritmica: l'algoritmo di Kruskal per trovare un albero di copertura di costo minimo [83], l'algoritmo di Dijkstra per trovare cammini minimi in un grafo, la versione di Edmond e Karp dell'algoritmo di Ford e Fulkerson per il problema del massimo flusso [50, 42], etc.

D'altra parte, per quanto riguarda la maggioranza dei problemi interessanti, compreso MIN TSP, sembrava che non ci fosse nessun modo di trovare efficientemente soluzioni ottime. All'inizio degli anni Settanta, la teoria della NP-completezza [32, 77, 85] fornì una spiegazione di questo fenomeno e una struttura teorica unificata per lo studio della *intrattabilità* dei problemi di ottimizzazione. Nell'arco di pochi anni, furono scoperti centinaia di risulati di NP-completezza (vedi [55] per una descrizione dello stato dell'arte nel 1979), e attualmente si stima che siano note alcune migliaia di problemi NP-completi, in massima parte (versioni decisionali di) problemi di ottimizzazione.

Essendo poco plausibile che i problemi di ottimizzazione NP-hard possano essere risolti efficientemente in modo ottimo, i progettisti di algoritmi sono forzati ad accontentarsi di mete più

[1] In questa tesi, come di consuto, diciamo che un algoritmo è *efficiente* se gira in tempo limitato da un polinomio nella dimensione dell'input.

modeste. In molti contesti, può essere estremamente utile avere a disposizione un algoritmo euristico che trovi soluzioni non ottime ma "sufficientemente buone". Per esempio, un algoritmo che risolva MIN TSP trovando cicli che siano al massimo l'1% più lunghi di un ciclo ottimo sarebbe altrettanto utile di un algoritmo ottimo in molte situazioni pratiche.

Quando esite una costante $r > 1$ tale che il costo della soluzione trovata dall'euristica è garantito essere entro un fattore moltiplicativo costante r dall'ottimo, allora l'euristica si dice un *algoritmo r-approssimato* (e r è chiamato il *rapporto di prestazione* dell'algoritmo).

Il progetto e l'analisi di algoritmi di approssimazione per problemi di ottimizzazione combinatoria è attualmente un campo vasto e tuttora in crescita dell'informatica teorica L'obiettivo finale di tale area è trovare, per ogni problema di ottimizzazione interessante, l' *algoritmo di approssimazione migliore possibile*, cioè l'algoritmo r-approssimate con il più piccolo valore di r.

Una tale impresa ha due aspetti: da una parte la ricerca algoritmica fornisce tecniche che danno algoritmi di approssimazione via via migliori; dall'altra, la teoria della complessità cerca di capire quali fattori di approssimazione siano *intrattabili* da ottenere e di interpretare risultati di *non-approssimabilità* in termini di completezza in classi di approssimazione.

Le due aree sono assai collegate, e capita che tecniche e risultati sviluppati per dimostrare risultati di complessità finiscano per migliorare algoritmi di approssimazione.

La presente tesi è soprattutto dedicata alla complessità dei problemi di approssimazione, anche se alcuni risultati hanno rilevanza algoritmica. Prima di descrivere i nostri risultati, presentiamo una breve storia del campo, in modo da mettere tali risultati nel giusto contesto.

1.1 Cenni storici

1.1.1 1973–1986: risultati iniziali

Gli algoritmi di approssimazione sono un modo molto naturale di affrontare la intrattabilità dei problemi di ottimizzazione e, anche se con una diversa terminologia, sono emersi inizialmente nell'area della ricerca operativa. Già nel 1996, un articolo di Graham [60] conteneva un'analisi nel caso peggiore della qualità delle soluzioni fornite da un algoritmo euristico per un problema di scheduling.

La definizione formale di algoritmo di approssimazione e di rapporto di prestazione è dovuta a un articolo pioneristico di Johnson [71]. In tale articolo, si accenna alla rilevanza di risultati di non-approssimabilità, viene presentato un algoritmo 2-approssimato per MAX SAT, un algoritmo $(1 + \ln n)$-approssimato per MIN SET COVER, e viene notato il fatto che alcuni approcci intuitivi per approssimare i problemi MAX CLIQUE e MIN NODE COLORING falliscono completamente. Il lavoro di Johnson ebbe una grande influenza: persino la notazione introdotta è, sostanzialmente, ancora usata. Alcune classi di approssimazione furono definite in tale articolo e poco dopo: APX (la classe dei problemi che sono r-approssimabili *per qualche* $r > 1$), PTAS la classe dei problemi che sono r-approssimabili *per ogni* $r > 1$), e FPTAS (la sottoclasse di PTAS che contiene i problemi che sono $(1 + \epsilon)$-approssimabili *per ogni* $\epsilon > 0$, in tempo polinomiale nella taglia dell'input e in $1/\epsilon$).

In pochi anni, l'area di ricerca progredì rapidamente sul campo algoritmico, ma emersero solamente pochi risultati di complessità.

L'unico strumento noto all'epoca per dimostrare risultati di intrattabilità era la teoria della NP-completezza, che studia *linguaggi*, cioè problemi decisionali. La complessità di risolvere in modo ottimo problemi di ottimizzazione può essere studiata nell'ambito della teoria della NP-completezza associando un linguaggio ad ogni problema. Sfortunatamente, si da il caso che i linguaggi associati a due problemi di ottimizzazione NP-hard siano computazionalmente equivalenti qualunque siano le proprietà di approssimabilità dei due problemi. Una considerevole attività di ricerca venne diretta al tentativo di superare questo inconveniente usando versioni modificate (più restrittive) della NP-completezza.

L'introduzione della NP-completezza *forte*, dovuta a Garey e Johnson [54], fu uno dei tentativi più riusciti. Quando un problema è fortemente NP-hard, purché alcune condizioni naturali siano soddisfatte, tale problema non può essere in FPTAS a meno che P = NP. In pochi anni, dimostrazioni di NP-completezza forti (quando possibili) divennero standard in articoli di complessità.

L'obiettivo successivo fu di trovare una restrizione della NP-completezza che potesse provare non-inclusione in PTAS. Una semplice osservazione è la seguente: supponiamo che, per una certa costante k, data una instanza x di un problema di ottimizzazione A, è NP-hard testare se l'ottimo è più grande di k o no. Un algoritmo r-approssimato può essere usato per rispondere ad una domanda del genere, purché $r < (k+1)/k$: ne consegue che A non può appartenere a PTAS a meno che P = NP. Alcuni risultati di non-approssimabilità sono stati dimostrati in questo modo, ma in genere sono piuttosto deboli. Una eccezione interessante è il problema dei k centri studiato da Hochbaum e Shmoys [66]. Tale problema può essere 2-approssimato, e la tecnica precedentemente descritta può essere usata per dimostrare che ottenere un rapporto di prestazione migliore di 2 è NP-hard. Per molti problemi di ottimizzazione interessante, come Max Clique, Min Vertex Cover e Max Sat, tale approccio è inutile perché, per ogni k fissato, è possibile verificare in tempo polinomiale se l'ottimo è maggiore di k o no.

Sahni e Gonzalez [102] usarono una tecnica di "incremento del gap" per dimostrare che Min TSP non è in APX. Questa tecnica funziona solo per problemi che non sono approssimabili entro alcun fattore.

Il risultato più interessante di questo periodo, dovuto a Garey e Johnson [53], è il fatto che Min Node Coloring non può essere r-approssimato per alcun $r < 2$. La loro dimostrazione parte dalla NP-completezza di 3-coloring (che già implica una non-approssimabilità entro $4/3 - \epsilon$), e poi il gap viene incrementato usando *prodotti di grafi*, una tecnica la cui potenza è stata ampiamente dimostrata di recente in diverse occasioni. Il prodotto di grafi è anche usato per dimostrare la *self-improvability* di Max Clique. Tale strana proprietà implica che o Max Clique è in PTAS, o non è in APX. Non era chiaro quale dei due casi valesse.

1.1.2 1987–1990: verso risultati più generali

Un'analisi dei primi risultati di non-approsimabilità rivela che quasi tutti si basano sulla complessità del problema decisionale associato al problema di ottimizzazione in questione. Un importante progresso nella teoria dell'approssimazione fu il rendersi conto che un diverso approccio era necessario: passare dalle versioni decisionali ai problemi di ottimizzazione medesimi; quindi definire *classi di problemi di ottimizzazione* e dimostrare risultati di *completezza* e di hardness in tali classi usando *riduzioni approximation-preserving.*

Già all'inizio degli anni Ottanta, l'idea di definire nozioni di riducibilità direttamente su problemi di ottimizzazione era stata concepita in due articoli pioneristici. Un articolo di Paz e Moran [98] è stato probabilmente il primo ad introdurre l'idea di riducibilità approximation-preserving, ma non contiene risultati di completezza in classi di approssimazione. Anche Ausiello, D'Atri e Protasi [8] definirono una nozione di riducibilità tra problemi di ottimizzazione. La loro riduzione preserva molta della struttura dei problemi (non le sole proprietà di approssimazione). Neanche in tale lavoro vi sono risultati di completezza.

Krentel [82] è stato il primo a dimostrare risultati di completezza in classi di problemi di ottimizzazione. La sua ricerca era principalmente orientata allo studio della complessità di calcolare soluzioni *ottime* per problemi di ottimizzazione. Per tale motivo, la riducibilità da lui definita non è approximation-preserving. I suoi risultati sono in parte sorprendenti, dal momento che dimostra che un problema in FPTAS (il problema KNAPSACK) e più difficile che MAX CLIQUE (in seguito, si dimostrerà che quest'ultimo è molto difficile da approssimare). Un modo di intepretare i risultati di Krentel è che la complessità di un problema di ottimizzazione non è collegata alla complessità della sua versione approssimata. Torneremo in seguito su questo punto e sul come esso sia collegato alla definizione corretta di riducibilità approximation-preserving.

Un articolo di Orponen e Manilla [92] contiene definizioni di riducibilità approximation-preserving e dimostrazioni di completezza in Min NPO[2]. Per la prima volta, viene osservato che un risultato di completezza in una classe di approssimazione (rispetto ad una riduzione approximation-preserving) implica un risultato di non-approssimabilità.

Crescenzi e Panconesi [35] estendono questa idea ad altre classi, e danno risultati di completezza in Max NPO e APX. La completezza in APX è una nozione particolarmente interessante. Un problema APX-complete non può appartenere a PTAS a meno che P = NP, e perciò la nozione di APX-completezza potrebbe essere uno strumento generale, simile alla NP-completezza, per dimostrare la non appartenenza a PTAS. Una volta stabilito il primo problema APX-completo, è sufficiente usare solamente riduzioni per provare nuovi risultati di completezza. Purtroppo, il problema APX-completo descritto in [35] è artificiale, e nessun problema completo naturale è stato trovato successivamente per molto tempo. È interessante notare che la APX-completezza di numerosi problemi naturali sia stata recentemente dimostrata *sfruttando precedenti risultati di non-approssimabilità*. Perciò, la completezza ha seguito la non-approssimabilità anziché viceversa. Torneremo su questo fatto nella prossima sezione.

Altri risultati di completezza furono ottenuti in seguito da Berman e Schnitger [22] e Kann [74] in classi di problemi con funzione obiettivo polinomialmente limitata.

Il lavoro di questo periodo che ha avuto la maggiore influenza sulla ricerca successiva è di Papadimitriou e Yannakakis [94], che introducono un approccio radicalmente diverso. Invece di cercare problemi completi in classi di approssimazione naturali (definite computazionalmente), si concentrano su problemi naturali e trovano una classe nella quale sono completi. Il loro punto di partenza è la caratterizzazione logica di NP data da Fagin [43]. Papadimitriou e Yannakakis ottengono una sottoclasse di APX introducendo una nozione di approssimazione nella caratterizzazione di Fagin, e chiamano tale classe Max NP. Essa comprende numerosi problemi, ad esempio MAX SAT, ma inizialmente non si riuscì a trovarvi alcun problema completo. Introducendo una restrizione nella sua definizione logica, si ottiene una classe più piccola, chiamata Max SNP, che contiene ancora problemi interessanti, come MAX CUT e MAX 3SAT, e che ha anche problemi completi naturali, inclusi MAX CUT e MAX 3SAT stessi. Altri problemi naturali, come MIN VERTEX COVER, sono invece Max SNP-hard. Tali risulati non risolvono la questione se MAX SAT e MIN VERTEX COVER sono in PTAS o no, ma unificano tali questioni in una sola: Max SNP è contenuto in PTAS? È chiaro che se esiste un problema in Max SNP che non è contenuto in PTAS, allora nessun problema Max SNP-hard può a sua volta appartenere a PTAS. A proposito della rilevanza di introdurre nuove classi per caratterizzare la complessità di problemi aperti, ci è gradito citare due bei passi da [94] e [96] sulla natura delle classi di complessità e sui motivi per cui vale la pena definirle.

> *In complexity, when we are faced with such a situation, in which a family of problems cannot be identified with the known complexity classes (...) one suspects that a* new complexity class *may be manifesting itself.*

[2] Vedi la Sezione 2.1 per una definizione di NPO, Max NPO, e Min NPO.

> *[Since these new classes] contain (...) natural, important, well studied complete problems, in some sense they existed before their definition, they are* discovered, *not* invented. *They represent computational paradigms* that must be collapsed with known ones, *if progress is to be made on understeing the complexity of the problems in he.*
>
> (C.H. Papadimitriou e M. Yannakakis)

Il successo di questo approccio fu vasto e immediato. Altri problemi Max SNP-hard furono rapidamente scoperti, tra i quali la restrizione di MIN TSP a spazi metrici [97], MIN STEINER TREE [23], shortest common superstring [25], multiway cuts [40].

Nel corso di una ricerca analoga, Berman e Schnitger [22] dimostrano che se esistono problemi in Max SNP che non ammettono un PTAS randomizzato, allora MAX CLIQUE non è approssimabile entro un fattore n^{ϵ}, per un certo $\epsilon > 0$. Il loro risultato viene dimostrato usando *prodotto di grafi randomizzato.*

L'approccio della definizione sintattica di classi di approssimazione con problemi completi naturali fu ulteriormente esplorato in una serie di articoli. Panconesi e Ranjan [93] dimostrano che MAX CLIQUE non è in Max SNP e definiscono una classa nella quale MAX CLIQUE è completo. Kolaitis e Thakur intraprendono una approfondita analisi delle classi sintattiche [81] e le estendono al caso dei problemi di minimizzazione. Ricerche successive hanno considerato il potere espressivo di vari formalismi logici in tale contesto, considerando anche operatori di punto fisso (vedi ad esempio [15]).

1.1.3 1991–1993: Il teorema PCP e le sue applicazioni

L'attività di ricerca che si sviluppò in seguito al lavoro di Papadimitriou e Yannakakis produsse numerosi risultati che collegavano tra loro le complessità di problemi diversi, ma nessun risultato che collegasse tali complessità a nozioni classiche come la NP-completezza. Il "collapse with known computational paradigms" era ancora di là da venire, e alla fine arrivò da una direzione del tutto inaspettata: la verifica probabilistica di dimostrazioni.

La verifica (deterministica) di dimostrazioni è inerente nella definizione standard di NP come la "classe dei linguaggi con dimostrazioni di appartenenza *brevi* e *facilmente verificabili*", dove *brevi* è inteso come *di taglia polinomiale* e *facilmente* sta per *fattibile in tempo polinomiale.* L'idea di estendere questo modello ammettendo anche *probabilismo* e *interazione* fu introdotta indipendentemente e simultaneamente da Goldwasser, Micali e Rackoff [59] e da Babai [9]. La classe dei linguaggi che ammettono tali sistemi di dimostrazione probabilistici interattivi fu chiamata IP. Ben-Or, Goldwasser, Kilian e Wigderson introdussero una estensione di questo modello nel quale il verificatore interagisce con due (o più) dimostratori (che non comunicano l'un l'altro); i linguaggi che ammettono questi stravaganti sistemi di dimostrazione formano la classe MIP. Fortnow, Rompel e Sipser [51] diedero una caratterizzazione alternativa di MIP: un linguaggio in MIP ammette dimostrazioni esponenzialmente lunghe che possono essere verificate da un verificatore polinomiale probabilistico[3]. Si supponeva che tali modelli fossero solo di poco più potenti di NP, fino a quando, Lund, Fortnow, Karloff e Nisan [88] dimostrarono che IP contiene la gerarchia polinomiale. Poco dopo, Shamir [103] dimostrò che IP = PSPACE e Babai, Fortnow e Lund [11] dimostrarono che MIP è uguale a NEXP. Quest'ultimo risultato era particolarmente sorprendente: il suo significato è che per ogni linguaggio che ammette prove di appartenenza esponenzialmente lunghe, tali prove possono essere verificate in tempo polinomiale con una piccola probabilità di errore. Il successivo passo

[3] In effetti, la terminologia di Fortnow, Rompel e Sipser era leggermente differente: facevano riferimento ad *oracoli* piuttosto che a *dimostrazioni.* Comunque non vi è alcuna differenza concettuale.

logico era quello di provare un teorema analogo per dimostrazioni più piccole, e quindi mostrare che ogni linguaggio in NP ha prove di appartenenza verificabil in tempo poli-logaritmico. Questo risultato fu ottenuto pochi mesi dopo da Babai, Fortnow, Levin e Szegedy [10]. Quasi simultaneamente, Feige, Goldwasser, Lovász, Safra e Szegedy [46] diedero un migliore sistema di dimostrazione per NP (con dimostrazioni verificabili in tempo $O(\log n \log\log n)$) e scoprirono la riduzione che avrebbe rivoluzionato il campo dell'approssimazione. Più specificamente, mostrarono come codificare una computazione del verificatore in un grafo, in un modo tale che ci fosse una corrispondenza tra cricche (cioè sottografi completi) nel grafo e presunte prove di appartenenza, e tra la taglia della crica e la probabilità che la prova di appartenenza sia accettata dal verificatore. In tal modo, un algoritmo di approssimazione per MAX CLIQUE può essere usato per approssimare la probabilità di accettazione del verificatore, e un gap tra la probabilità di accettazione di dimostrazioni giuste e sbagliate si traduce in un risultato di non-approssimabilità per MAX CLIQUE. Assumendo che NP non ammetta algoritmi sub-esponenziali, la conseguenza è che MAX CLIQUE non è approssimabile entro un fatore $2^{\log^{1-\epsilon} n}$ per alcun $\epsilon > 0$. La connessione tra verifica di dimostrazioni e approssimazione fu anche introdotta in un articolo di Condon [31].

Successivamente, il progresso fu veloce. In un lavoro fondamentale, Arora e Safra [6] notarono che per migliorare il risultato di [46] non era necessario limitare il tempo di calcolo del verificatore, ma solo il numero di *bit casuali* usati e di *bit letti* dalla dimostrazione. Arora e Safra introdussero la notazione $\mathsf{PCP}(r(n), q(n))$ che denota la classe dei linguaggi che hanno prove di appartenenza che possono essere verificate probabilisticamente usando $O(r(n))$ bit casuali e $O(q(n))$ bit di lettura. Infine, ridussero il numero di bit di lettura introducendo e sfruttando la nuova idea di *comporre ricorsivamente* verificatori: un verificatore verifica la prova che un verificatore accetta quando verifica la prova che un verificatore accetta quando ...Il loro risultato principale fu $\mathsf{NP} = \mathsf{PCP}(\log n, \sqrt{\log n})$, che implica la NP-hardness di approssimare MAX CLIQUE entro un fattore $2^{\sqrt{\log n}}$.

Il passo successivo fu il rendersi conto che la complessità di altri problemi di ottimizzazione poteva essere stabilita usando il modello PCP. Sudan e Szegedy osservarono che se $\mathsf{NP} = \mathsf{PCP}(\log n, 1)$, allora $\mathsf{Max\ SNP}$ non è contenuto in PTAS (a meno che $\mathsf{P} = \mathsf{NP}$). Poche settimane dopo, il risultato $\mathsf{NP} = \mathsf{PCP}(\log n, 1)$ fu ottenuto da Arora, Lund, Motwani, Sudan e Szegedy [5]: tale risultato è noto attualmente come il *teorema PCP*. Altre all'applicazione ai problemi $\mathsf{Max\ SNP}$-hard, il teorema PCP implica anche che è NP-hard approssimare MAX CLIQUEentro un fattore n^{ϵ}, per un certo $\epsilon > 0$.

Si dimostrò poi che anche altri problemi di ottimizzazione sono collegati alla verifica di prove, e un gran numero di questioni che erano aperte da molto tempo furono rapidamente risolte una dopo l'altra. In particolare, Lund e Yannakakis [89] dimostrarono che MIN SET COVER non può essere approssimato entro $1/4 \log n$ (un valore dello stesso ordine di grandezza della prestazione dell'algoritmo di Johnson)e che MIN NODE COLORING non è approssimabile entro n^{ϵ} per un certo $\epsilon > 0$. Queste riduzioni non sono dal verificatore del teorema PCP, ma dal modello leggermente differente dei sistemi di dimostrazione con due dimostratori e un round [84, 49] (Bellare [17] fu forse il primo e esplorare la connessione tra tali sistemi di dimostrazione e problemi di approssimazione).

1.1.4 Risultati contemporanei

La scoperta di nuovi risultati in questo campo ha preso un ritmo molto rapido negli ultimi tre anni. Anche articoli di rassegna molto recenti risultano essere leggermente datati [18, 75].

I miglioramenti nei risultati di non-approssimabilità rappresentano forse le conquiste recenti più sorprendenti. Tali risultati si basano su versioni migliorate del teorema PCP e su nuove riduzioni.

Recentemente, risultati di non approssimabilità *tight* sono stati stabiliti per una serie di importanti problemi di ottimizzazione.

D'altra parte, la grande potenza delle tecniche PCP ha dato nuovo impulso alla teoria *strutturale* (cioè basata su classi, riduzioni e completezza) dell'approssimazione.

Risultati di non-approssimabilità

Tutti i risultati di non-approssimabilità ottenuti dalla prim versione del teorema PCP [5] sono di tipo "esistenziale": per esempio si può dimostrare che esistono costanti ϵ_s e ϵ_c tali che Max Sat non può essere approssimato entro $1 + \epsilon_s$ e Max Clique non può essere approssimato entro n^{ϵ_c}, ma queste costanti non vengono esplicitamente stimate.

Una versione ottimizzata del teorema PCP fu dimostrata da Bellare, Goldwasser, Lund e Russel [20]. Il loro risultato implica che Max Clique non può essere approssimato entro $n^{1/30-\epsilon}$ per ogni $\epsilon > 0$ (d'ora in avanti, ϵ denoterà sempre una qualche costante fissata arbitrariamente piccola). Il progresso nella stima della costrante fu veloce. Il risultato di non-approssimabilità per Max Clique era stato ottenuto costruendo un verificatore PCP per NP che leggeva solamente 29 bit, e usando un risultato di Zuckerman [111] (basato sul prodotto randomizzato di grafi di [22]) che diceva che se i linguaggi in NP possono essere verificati leggendo q bit, allora è difficile approssimare Max Clique entro $n^{1/(q+1)-\epsilon}$. Successivamente, Feige e Kilian [47] osservarono che si poteva usare un parametro più raffinato del numero di bit letti: se un verificatore usa f *free bit* allora Max Clique non è approssimabile entro $n^{1/(f+1)-\epsilon}$. Intuitivamente, un verificatore usa f free bit se, una volta letti i suoi bit casuali e richiesti i bit della prova in lettura, ci sono al più 2^f possibili esiti di tale lettura che lo facciano accettare. Dunque, i free bit sono sempre non più dei bit letti (dopo aver richiesto q bit, non ci possono essere più di 2^q possibili risposte) e sono di solito molto meno. Questa nuova idea implicava un risultato di non-approssimabilità $n^{1/15-\epsilon}$ per Max Clique. Un ulteriore raffinamento du proposto da Bellare e Sudan [21], che introdussero la nozione di *free bit ammortizzato*: un verificatore usa $\bar{f}$ free bit ammortizzati se usa f free bit, accetta con probabilità uno prove corrette, accetta con probabilità al più s prove sbagliate, e $\bar{f} = f/\log(1/s)$. Questa nozione di "ammortizzazione" cerca di catturare il fatto che si è disposti ad usare più free bit se questo comporta una considerevole dimimuzione della probabilità di errore s. In effetti, se i linguaggi in NP possono essere verificati con $\bar{f}$ free bit ammortizzati, ne consegue che Max Clique non è approssimabile entro $n^{1/(\bar{f}+1)-\epsilon}$. Bellare e Sudan mostrarono che $3+\epsilon$ free bit ammortizzati sono sufficienti, e perciò Max Clique è difficile da approssimare entro $n^{1/4-\epsilon}$. Bellare, Goldreich e Sudan [19] svilupparono successivamente un sistema che usava $2 + \epsilon$ free bit ammortizzati, poi Håstad mostrò che $1+\epsilon$ erano sufficienti [62] e, infine, che ϵ free bit ammortizzati erano sufficienti [61]. Dunque, per ogni $\epsilon > 0$, è difficile approssimare Max Clique entro $n^{1-\epsilon}$.

Un simile continuo miglioramento fu ottenuto per Min Node Coloring. Il risultato finale dovuto a Feige e Kilian [48] è che, per ogni $\epsilon > 0$, è difficile approssimare Min Node Coloring entro $n^{1-\epsilon}$. Il risultato di Feige e Kilian si basa sul verificatore con ϵ free bit ammortizzati di Håstad.

Il risultato finale per Min Set Cover è di Feige [44], che ha mostrato che è difficile approssimare tale problema entro $(1-\epsilon)\ln n$ (si ricordi che l'algoritmo di Johnson aveva un rapporto di prestazione $\ln n + 1$).

Il primo risultato di non-approssimabilità per Max Sat con una costante esplicita è stato stabilito da Bellare, Goldwasser, Lund e Russel [20], a partire da una versione migliorata del teorema PCP che comprendeva una descrizione esplicita delle computazioni del verificatore e un limite inferiore esplicito sulla probabilità di accettare prove sbagliate (*soundness*); prove corrette erano accettate con probabilità uno. Una volta codificate le computazioni del verificatore usando piccole formule CNF (così come in [5]), il risultato di non-approssimabilità per Max Sat che ne consegue può essere calcolato come funzione della taglia delle formule CNF e della soundness. Il risultato provato in [20] era una non-approssimabilità entro 94/93. Ulteriori miglioramenti furono ottenuti usando lo stesso approccio, ma partendo da versioni migliori del teorema PCP (cioè con computazioni più semplici e/o con una soundness più piccola). Bellare e Sudan [21] dimostrarono un fattore di non-approssimabilità 73/72, mentre Bellare, Goldreich e Sudan [19] arrivarono a 27/26: il miglior risultato attuale (forse il migliore possibile) è $8/7 - \epsilon$, docuto ad Håstad [63].

Risultati strutturali

Una nuova serie di risultati strutturali è stata resa recentemente possibile dal teorema PCP e dai suoi potenti corollari.

Khanna, Motwani, Sudan, e Vazirani [80] usano il teorema PCP per dimostrare che Max 3Sat è completo in APX PB (la restrizione di APX a problemi con funzione obiettivo polinomialmente limitata). Più in generale, introducono una nuova tecnica di riduzione che consente di ottenere un risultato di completezza a partire da un risultato di non-approssimabiltià, purché il problema in questione possegga una certa proprietà di "additività". QUesto risultato chiude una serie di importanti questioni aperte nella teoria strutturale dell'approssimazione, compresa la questione de Max NP ha problemi completi e se Max Sat ammette una riduzione approximation-preserving a Max 3Sat (la risposta è affermativa in entrambi i casi). Questa tecnica di riduzione, pur molto utile e generale, funziona solo all'interno di classi di approssimazione con funzione obiettivo polinomialmente limitata.

Una linea di ricerca differente è stata intrapresa da Chang, Gasarch e Lund [29, 28], che usano il teorema PCP per investigare la *query complexity* di problemi di approssimazione. Per esempio, il risultato di Håstad [63] implica che è NP-hard approssimare Max Sat entro un fattore 1.142. Dunque, facendo girare una volta un algoritmo 1.142-approssimato per Max Sat è possibile rispondere ad una domanda NP. È anche possibile rispondere a due o più domande? Chang, Gasarch e Lund esplorano questa idea e danno limitazioni inferiori e superiori al numero di domande NP che sono codificabili in una istanza di un problema di approssimazione. Una conseguenza dei loro risultati è che calcolare una soluzione 2-approssimata per il numero di cricca di un grafo è strettamente più facile che calcolare il numero di cricca esatto. Anche se i due problemi sono entrambi NP-hard, il secondo non può essere ridotto al primo a meno che P = NP.

1.2 I risultati di questa tesi

Diamo adesso una sintesi dei risultati originali contenuti in questa tesi. Vi sono due principali gruppi di risultati: uno riguardante le proprietà di riduzioni e completezza in classi di approssimazione, e uno riguardante nuovi risultati di non-approssimabilità.

Temi unificanti sono lo sviluppo di nuovi paradigmi per trovare *riduzioni approximation-preserving* e l'enfasi su *risulati generali* che possano essere applicati a numerosi *problemi naturali.*

Alcune idee di base riaffiorano in diversi contesti, come ad esempio la tecnica di *scaling* per ridurre un problema generale a un problema *polinomialmente limitato.* L'introduzione di un nuovo tipo di riduzione approximation-preserving risolve alcune questioni aperte nella teoria strutturale dell'approssimazione, ma ha anche un'interessante applicazione algoritmica.

1.2.1 La struttura della classi di approssimazione

Riduzioni e completezza

La ricerca sulla teoria strutturale dell'approssimazione e sulla non-approssimabilità ha prodotto numerosi risultati di *hardness* negli ultimi anni. Per vedere tali risultati come risultati di *completezza* era necesserio introdurre molte classi, talvolta dalla definizione alquanto artificiale. Il lavoro di Khanna, Motwani, Sudan e Vazirani [80] ha unificato molte di queste classi; tuttavia, abbiamo già menzionato il fatto che la loro tecnica non si estende al di là di classi polinomialmente limitate. Per di più, in alcune classi si era ancora costretti a distinguere tra problemi di massimizzazione di minimizzazione: c'erano problemi Max NPO-completi e Min NPO-completi ma nessun problema NPO-completo.

Una ulteriore fonte di complicazioni era costituito dal gran numero di differenti riducibilità approximation-preserving apparse nella letteratura. Alcune di loro non sembrano essere confrontabili, perciò un risultato di completezza o di hardness dimostrato rispetto ad una riducibilità potrebbe non valere rispetto ad un'altra.

Presentiamo in questa tesi uno *studio sistematico* delle proprietà delle riducibilità approximation-preserving conosciute. Dimostriamo che la mancanza di risultati di completezza unificati in un piccolo numero di classi è dovuta a *limitazioni inerenti* delle riducibilità note. In particolare, dimostriamo che:

1. La L-riducibilità usata da Papadimitriou e Yannakakis [94] non conserva l'approssimabilità entro un fattore costante. Inoltre, la chiusura di Max SNP rispetto a L-riduzioni contiene problemi non approssimabili, Entrambi i risultati sono dimostrati assumendo $\mathsf{P} \neq \mathsf{NP} \cap co\text{-}\mathsf{NP}$.

2. Max Sat non è APX-completo rispetto alla E-riducibilità (usata da Khanna e altri [80]) né rispetto alla P-riducibilità (usata da Crescenzi e Panconesi [35]), a meno che $\mathsf{P}^{\mathsf{NP}} = \mathsf{P}^{\mathsf{NP}[O(\log n)]}$ (un evento ritenuto poco plausibile).

3. Max Sat non è nemmeno APX PB-completo rispetto alla L-riducibilità, a meno che $\mathsf{P} = \mathsf{NP}$.

Il primo risultato contraddice una diffusa convinzione. Il modo in cui dimostriamo il punto 2 indica quale sia la limitazione delle attuali riducibilità: non solo conservano l'approssimabilità, ma anche la difficoltà di calcolare soluzioni ottime. Ciò è forse un retaggio dei vecchi approxxi basati sulla NP-completezza. Come dimostrato da Krentel [82], però, un problema che sia più facile di un altro da approssimare, può essere più difficile da risolvere in modo ottimo. Una riducibilità che conservi simultaneamente la difficoltà di risolvere i problemi in modo ottimo e approssimato non può quindi evitare di manifestare dei comportamenti patologici in certe circostanze.

Contributo innovativo. *Sfruttiamo strumenti avanzati di* teoria strutturale della complessità*, mai usati prima nell'analisi di riduzioni approximation-preserving.*

Per superare tale limitazione, proponiamo la definizione di una nuovo nozione di riducibilità, che chiamiamo *AP-riducibilità*, la quale conserva l'approssimazione ma non *la risolubilità ottimale*, Rispetto alla AP-riducibilità, dimostriamo:

1. L'esistenza di problemi naturali NPO-completi, unificando le precedenti classi di problemi Max NPO- e Min NPO-completi.

2. Analogamente, l'esistenza di problemi naturali NPO PB-completi.

3. L'esistenza di problemi naturali APX-, logAPX- e polyAPX-completi, superando la precedente restrizione a classi polinomialmente limitate.

CONTRIBUTO INNOVATIVO. *La novità principale nella definizione di AP-riducibilità è di* conoscere *il fattore di approssimazione che si vuole conservare.*

La APX- e polyAPX-completezza di problemi naturali è dimostrata con tecniche che si basano sul teorema PCP. Introduciamo una tecnica di riduzione basata sul metodo probabilistico che consente di dimostrare direttamente che MAX 3SAT è Max NP-complete.

L'idea di base del nostro approccio è meglio spiegata nel caso particolare della riduzione da MAX SAT a MAX 3SAT. Le tecniche standard di riduzione, basate su sostituzioni locali, non consentono di ridurre MAX SAT a MAX 3SAT a causa della possibile presenza di clasusole grandi; ma le clausole grandi sono facili da soddisfare usando un semplice algoritmo probabilistico (che, a sua volta, ha scarse prestazioni con clausole piccole). Il nostro metodo consiste nel *combinare* (in un senso probabilistico) una soluzione basata su riduzioni standard e una soluzion data dall'algoritmo probabilistico: questa soluzione "mista" sarà buona per qualunque combinazione di clausole grandi e piccole.

CONTRIBUTO INNOVATIVO. *L'idea di combinare probabilisticamente soluzioni differenti è stata usata per ottenere* algoritmi di approssimazione *(ad esempio in [57]) ma noi la usiamo per la prima volta per ottenere una* riduzione approximation-preserving.

La Max NP-completezza di MAX 3SAT è un semplice corollario della sua APX-completezza, ma la nostra riduzione ha una rilevanza sia *strutturale* sia *algoritmica* a causa del fatto di non fare uso di tecniche PCP.

Da un punto di vista *strutturale*, la Max NP-completezza di MAX 3SAT significa che i problemi Max NP non sono più difficili da approssimare di quelli Max SNP. Nella definizione di Max SNP (vedi anche [94] e la Sezione 2.1) c'è una struttura logica più ricca che in quella di Max SNP, e la riduzione che usa il teorema PCP non spiega chiaramente come sia possibile superare questa differenza. Ciò è dovuto al fatto che le riduzioni PCP sono fortemente *globali* in un senso molto forte, e nel ridurre un problema logica ad un altro, non è chiaro quale variabile sia mappata in quale e quale vincolo sia mappato in quale, e così manca una chisara relazione tra la struttura del problema di partenza e quella del problema di arrivo. Al contrario, la nostra riduzione da una spiegazione molto semplice della relazione tra Max NP e Max SNP: la differenza essenziale tra le due classi è nella definizione logica dei problemi Max NP sono ammesse disgiunzioni arbitrariamente grandi. Ma grandi disgiunzioni sono facili da soddisfare (in un senso probabilistico) e perciò non possono rendere un problema più difficile.

Una riduzione approximation-preserving da un problema A ad un problema B può implicare uno *schema di approssimazione* nel caso seguente. Assumiamo che il problema B ammetta uno schema di approssimazione quando venga ristretto ad un certo sottoinsieme $\mathcal{I}_B$ delle sue istanze, e che sia possibile trovare un insieme $\mathcal{I}_A$ di istanze di A tali che ciascuna di loro sia mappata dalla riduzione in una istanza di $\mathcal{I}_B$. Allora, la riduzione, insieme allo schema di approssimazione da uno schema di approssimazione per le istanze di $\mathcal{I}_A$. La nostra riduzione, in effetti, implica uno schema di approssimazione per la restrizione *planare* di MAX SAT e MAX GSAT-B. Tale schema di approssimazione gira in tempo lineare, mentre schemi precedenti richiedevano tempo $n^{O(1/\epsilon)}$. Un tale risultato sarebbe stato impossibile usando riduzioni PCP, per almeno due motivi:

1. È molto difficile caratterizzare i risultati di una riduzione PCP, e capire così quali insiemi di istanze saranno mappati in quali.

2. Le riduzioni PCP generano sempre istanze che sono difficili da approssimare, anche quando l'istanza di partenza ha una stuttura molto semplice.

Le considerazioni precedenti suggeriscono che una dimostrazione senza PCP che Max 3Sat è APX-completo avrebbe conseguenze molto interessanti sia sul piano algoritmico sia su quello strutturale. Dimostreremo però che, purtroppo, almeno una versione più debole del teorema PCP è necessaria per dimostrare la APX-completezza di Max 3Sat.

La "Query Complexity" e le sue applicazioni

L'argomento successivo trattato in questa tesi è l'analisi delle proprietà dei problemi completi in classi di approssimazione. Ci concentriamo sulla query complexity, che è stata già usata per misurare la complessità di problemi di approssimazione, ma che noi adattiamo per la prima volta allo studio della complessità di *calcolare soluzioni approssimate* (mentre studi precedenti avevano considerato la complessità di *valutare l'ottimo in modo approssimato*).

Diamo una caratterizzazione della APX-completezza e della polyAPX-completezza in termini di query complexity e proprietà combinatorie. I nostri risultati mostrano che la query complexity non è solo un *effetto* della complessità dell'approssimazione, ma che è sostanzialmente lo *stesso fenomeno*, osservato da una prospettiva differente. Ciò spiega anche un fenomeno paradossale menzionato in precedenza: nella teoria della complessità, di solito, l'intrattabilità di un problema (assumendo congetture standard, come $\mathsf{P} \neq \mathsf{NP}$) viene dimostrata come corollario di un risultato di completezza in una classe di complessità. In APX, le cose sono andate nel verso opposto: la completezza di problemi naturali, come Max Sat, è stata dimostrata usando il teorema PCP, vale a dire, basandosi sulla intrattabilità dell'approssimazione. La nostra caratterizzazione della APX-completezza ci dive che per i problemi APX che godano di una certa proprietà di "additività" (incluso Max Sat) l'intrattabilità dell'approssimazione è equivalente alla APX-completezza.

I nostri risultati sulla query complexity hanno anche una serie di implicazioni sulle proprietà di approssimazione di problemi naturali.

1. I problemi Min Bin Packing, Min Edge Coloring, e altri, sono APX-intermedi, cioè non sono in PTAS (a meno che $\mathsf{P} = \mathsf{NP}$, e questo era ben noto) né sono APX-completi (a meno che la gerarchia polinomiale collassi, questo è il nostro risultato).

2. I problemi Min Set Cover, Min Dominating Set e Min Hitting Set non sono self-improvable a meno che la gerarchia polinomiale collassi.

3. Il problema di calcolare una soluzione ottima per il problema Max Clique non può essere ridotto al problema di 2-approssimare Max Clique, a meno che la gerarchia polinomiale collassi.

Ci è gradito fare alcuni commenti sui nostri risultati. Riguardo il primo, si tratta del primo risultato di "intermedietà" dimostrato per problemi naturali in classi di complessità standard. Se $\mathsf{NP} = co\text{-}\mathsf{NP}$, allora i problemi elencati al punto (1) in precedenza sono tutti APX-completi. Ciò suggerisce che il nostro risultato non possa essere dimostrato sotto l'ipotesi $\mathsf{P} \neq \mathsf{NP}$. C'è tuttavia ancora un certo gap tra la condizione necessaria e quella sufficiente. Sotto la nostra definizione (molto debole) di self-improvability, Min Set Cover, Min Dominating Set, e Min Hitting Set sono self-improvable se $\mathsf{NP} = co\text{-}\mathsf{NP}$. Di nuovo, c'è un certo gap tra la condizione necessaria e quella sufficiente. Non sappiamo come dimostrare i nostri risultati del punto (2) senza usare la query complexity, ed è strano che un tale specialistico concetto di complessità strutturale abbia qualcosa a che fare con una fondamentale proprietà algoritmica come la self-improvability. Il terzo risultato risolve una questione lasciata aperta da Chang [28].

CONTRIBUTO INNOVATIVO. *Estendiamo l'approccio della query complexity al problema di calcolare soluzioni approssimate per un problema di ottimizzazione. Risultati precedenti [29, 28] si limitavano a considerare il problema di valutare in modo approssimato l'ottimo.*

1.2.2 Nuovi risultati di non-approssimabilità

Il metodo attualmente standard per dimostrare risultati di non-approssimabilità consiste nel partire da una versione ottimizzata del teorema PCP, e poi nel mostrare come codificare le computazioni del verificatore in una istanza del problema in questione, in modo tale che l'ottimo di tale istanza rifletta la probabilità di accettazione del verificatore. La riduzione FGLSS [46] e la riduzione in [5] sono, da questo punto di vista, paradigmatiche.

Migliori riduzioni locali

I *problemi di soddisfacimento di vincoli* sono generalizzazioni del problema MAX SAT. L'istanza è un insieme di "clausole" (o *vincoli*) definite su variabili booleane: tali clausole non sono necessariamente disgiunzioni di letterali, ma possono essere funzioni booleane arbitrarie (problemi diversi sono definiti usando diversi insiemi di funzioni booleane ammissibili). L'obiettivo è massimizzare il numero di clausole soddisfatte o di minimizzare il numero di clausole contraddette. Oltre a MAX SAT e alle sue restrizioni, numerosi problemi naturali rientrano in questo contesto, compresi problemi su grafi come MAX CUT e MAX DICUT.

Per dimostrare un risultato di non-approssimabilità per un problema di massimo soddisfacimento di vincoli si usa una riduzione per *sostituzione locale* dal verificatore del teorema PCP. Questo approccio è standard per MAX SAT (sostanzialmente, risale a [5]) ma è stato introdotto solo recentemente per altri problemi da Bellare, Goldreich e Sudan [19]. Per catturare riduzioni differenti in un solo schema generale, essi formalizzano la nozione di *gadget* che viene usato in riduzioni per sostituzione locale. Informalmente, un gadget è una struttura combinatoria che trasforma un vincolo di una problema di ottimizzazione in vincoli di una altro problema. Bellare, Goldreich e Sudan associano anche un *costo* ad ogni gagdet: meno costoso è il gadget, migliore è la riduzione (e il conseguente risultato di non-approssimabilità). Ne consegue che una migliore costruzione di gadget comporta un miglior risultato di non-approssimabilità. Per questo motivo, è desiderabile avere *limitazioni inferiori* sul costo di certi gadget, per capire se sia possibile migliorare le costruzioni note o se esse siano le migliori possibili. La questione di trovare limitazioni inferiori sul costo di gadget è stata posta in [19].

La costruzione di questi oggetti combinatori sembra richiedere ogni volta idee nuove, e il lavoro deve essere fatto, partendo essenzialmente da zero, ogni volta che viene introdotto un nuovo verificatore PCP che usi test di tipo diverso. Per esempio, il verificatore descritto nella prima e nella seconda versione di [19] è diverso sa quello descritto nella terza versione. È stato necessario trovare nuovi gadget per provare risultati di non-approssimabilità con il nuovo verificatore.

Mostreremo in questa tesi come risolvere completamente tali problemi attraverso una tecnica basata su programmazione lineare (PL) che consente di trovare *automaticamente* gadget che siano *dimostrabilmente ottimi.*

La nostra tecnica dimostra che i gadget noti per MAX 2SAT e MAX 3SAT (introdotti in [19]) sono i migliori possibili. Il gadget per MAX CUT non è ottimo, e ne troviamo uno ottimo costruito al calcolatore, producendo così un miglior risultato di non-approssimabilità per MAX CUT.

Problema	Non-approssimabilità	Migliore approssimazione
Max 2Sat	1.047 [19, 63]	1.074 [45]
Max 3Sat	1.142 [63]	**1.249** (1.299 [57, 58, 45])
Max Sat	1.142 [63]	1.319 [57, 58]
Max Cut	**1.062** (1.055 [19, 63])	1.139 [58]
Max DiCut	**1.083** (1.055 [19, 63])	1.169 [45]
Max 2CSP	**1.111** (1.047 [19, 63])	1.169 [45]
Max 3CSP	$2-\epsilon$ [63]	**2.721** (4 [105], 7 [3])
Min Vertex Cover	1.166 [63]	2 [56]

Tabella 1.1: Risultati di approssimabilità e di non-approssimabilità per alcuni problemi di soddisfacimento di vincoli. Riportiamo in grassetto i nostri risultati, e tra parentesi i fattori che sarebbero stati ottenibili senza usare i risultati di questa tesi.

Presentiamo anche gadget ottimi (e migliori risultati di non-approssimabilità) per Max DiCut e Max 2CSP. I nostri risultati sono stati ottenuti quando il miglior verificatore sul mercato era quello di Bellare et al. [19]. Usando il loro verificatore e la nostra riduzione, abbiamo dimostrato che Max Cut (rispettivamente, Max DiCut) era NP-hard da approssimare entro 61/60 (rispettivamente, 45/44): per entrambi i problemi, la non-approssimabilità dimostrata in [19] era 72/71[4]. Usando il veirificatore di Håstad [63], dimostriamo un fattore di non-approssimabilità $17/16-\epsilon$ (rispettivamente, $13/12-\epsilon$). Usando il verificatore di Håstad insieme ai gadget di [19] si sarebbe ottenuto un fattore $19/18-\epsilon$ per entrambi i problemi.

Ottenere migliori riduzioni tra problemi può anche dare come risultato migliori algoritmi di approssimazione. Illustriamo questo punto costruendo un gagdet che riduce Max 3Sat a Max 2Sat. Usando questa nuova riduzione insieme ad una tecnica di Goemans e Williamson [58] l'algoritmo 1.074-approssimato per Max 2Sat di Feige e Goemans [45], otteniamo un algoritmo 1.249-approssimato per Max 3Sat. Il miglior risultato che poteva essere ottenuto precedentemente , combinando la tecnica di [57, 58] e l'algoritmo di [45], era 1.299 (vedi anche l'algoritmo 1.301-approssimato di Asano, Hirata e Ono [91]). Analogamente, costruiamo un gadget che riduce Max 3CSP a Max 2Sat e, usando l'algoritmo di Feige e Goemans, otteniamo un algoritmo 2.721-approssimato per Max 3CSP. Il miglior risultato precedente per questo problema era un fattore di approssimazione 4 [105].

I nostri risultati, e alcuni altri risultati collegati, sono riassunti nella Tabella 1.1.

Infine, la formulazione PL consente di dimostrare limitazioni inferiori usando l'analisi duale. Applichiamo questa tecnica in un caso nel quale una analisi automatica non sarebbe stata possibile (dato che dimostriamo limitazioni inferiori per un numero *infinito* di problemi).

Contributo innovativo. *Il passaggio principale della nostra costruzione è un lemma combinatorio che afferma che esistono gagdet ottimi che soddisfano una certa forma normale. La dimostrazione sfrutta l'idea di usare* sostituzioni *di variabili in modo da trasformare ogni gadget in uno in forma normale senza incrementare il costo. L'osservazione successiva è che la spazio di ricerca formato dai gadget in forma normale può essere descritto tramite disuguaglianze lineari, e il costo del gadget è una funzione lineare. Pertanto, trovare il gadget migliore si riconduce a risolvere un problema PL, che può essere automaticamente risolto in modo ottimo. La teoria della* dualità *in programmazione lineare può essere usata per dimostrare l'ottimalità dei gadget costruiti in questo modo* senza che si debba assumere la correttezza *del programma che esegue il calcolo del PL.*

[4] Il risultato menzionato in [19] è in realtà 66/65, ma ciò è dovuto ad un errore di calcolo scoperto da Håstad.

Eliminare i pesi

I risultati di non-approssimabilità che si dimostrano tramite riduzioni con gadget valgono per problemi pesati (è sufficiente però restringersi a considerare pesi polinomialmente limitati). Si noti che un problema di soddisfacimento di vincoli con pesi polinomialmente limitati può essere visto come un problema in cui ammette la ripetizione di clausole. I migliori algoritmi noti per tali problemi funzionano altrettanto bene per la versione non pesata e per quella pesata (anche se i pesi non sono polinomialmente limitati).

Per il problema Max Cut in grafi semplici (cioè senza pesi sugli archi e senza archi multipli), Bellare, Goldreich e Sudan [19] hanno dimostrato un risultato di non-approssimabilità usando una riduzione dalla versione pesata del problema. La loro riduzione elimina i pesi dal problema ma indebolisce il risultato di non-approssimabilità: applicando tale riduzione al nostro risultato per Max Cut implicherebbe un fattore di non-approssimabilità circa 1.02 per Max Cut non pesato. Risultati di non-approssimabilità con costanti esplicite per Max Sat senza ripetizione di clausole non erano noti.

Una ragionevole congettura che potrebbe essere estratta da questa situazione è che le versioni non pesate di Max Cut, Max Sat etc. sono più facili da approssimare che le rispettive versioni pesate. Il fatto che gli algoritmi noti si comportino altrettanto bene nei due casi potrebbe spiegarsi osservando che tutti quanti usano programmazione lineare o semidefinita. Forse, nuove tecniche algoritmiche potrebbero migliorare l'approssimazione nel caso non pesato.

Dimostreremo invece che per numerosi problemi di soddisfacimento di vincoli (inclusi Max Cut, Max DiCut, Max 2Sat, Max E3Sat etc.) la versione pesata è esattamente altrettanto difficile da approssimare della versione non pesata. Più formalmente, un algoritmo r-approssimato per la versione non pesata può essere trasformato in un algoritmo $(r + o(1))$-approssimato per la versione pesata.

Da una parte, i nostri risultati forniscono una spiegazione del comportamento di algoritmi noti, chiarificando che l'avere un'approssimazione altrettanto buona nel caso pesato e in quello non pesato è intrinseco nel problema e non è dovuto ad una particolare tecnica algoritmica. D'altra parte, otteniamo i migliori risultati di non-approssimabilità per le versioni non pesate di questi problemi. Ogni futuro risultato di non-approssimabilità per la versione pesata si applicherà immediatamente anche alla versione non pesata.

Contributo innovativo. *Per il problema* Max Cut*, la nostra riduzione funziona sostituendo un insieme di nodi al posto di un nodo e gli archi di un expander bipartito al posto di un arco pesato. La trasformazione di un taglio nel grafo non pesato in un taglio nel grafo pesato usa il* metodo probabilistico*; la dimostrazione che le due soluzioni hanno circa lo stesso rapporto di prestazione sfrutta la proprietà* mixing *degli expander bipartiti. La generalizzazione ad altri problemi di soddisfacimento di vincoli richiede l'introduzione (e la costruzione esplicita) di* insiemi mixing di tuple.

Il problema vertex cover

Il primo risultato esplicito di non-approssimabilità per Min Vertex Cover è dovuto a Bellare, Goldreich e Sudan [19], che usano la riduzione di [46] combinandola con la riduzione standard da Max Clique a Min Vertex Cover. Il risultato da loro dimostrato è un fattore di non-approssimabilità 17/16, portato a $7/6 - \epsilon$ da Håstad [63].

Dimostriamo in questa tesi che Min Vertex Cover può essere visto come un problema di minimo soddisfacimento di vincoli: è equivalente ad un problema noto come Min Sat (questo risultato è stato scoperto indipendentemente in [90] e in [65]). Questa equivalenza fornisce un miglior algoritmo di approssimazione e un nuovo risultato di non-approssimabilità per Min Sat.

Così come per i problemi di massimizzazione trattati in precedenza, siamo anche interessati a confrontare le versioni pesate e non pesate di Min Vertex Cover e di Min Sat. In questo caso, una dimostrazione più semplice sarà sufficiente per mostrare che sono altrettanto difficili da approssimare.

Un'importante restrizione di Min Vertex Cover è quella a grafi di grado limitato. Anche tale versione ristretta rimane APX-completa, anche se il grado è limitato da tre. Questa restrizione di Min Vertex Cover è usata per dimostrare la APX-completezza di altri problemi, inclusi il problema della sottosequenza comune più corta e Min Steiner Tree [25, 23]. La dimostrazione di APX-completezza per Min Vertex Cover-B usa una riduzione da Max 3Sat. Il fattore di non-approssimabilità peggiora molto nel corso della riduzione (circa 200 volte).

Presentiamo in questa tesi una riduzione alternativa che produce grafi di grado grande (ma limitato da una gerta costante) e che peggiora il fattore di non-approssimabilità in modo trascurabile. In particolare, per una certa costante B, è NP-hard approssimare Min Vertex Cover-B entro 1.16. Si noti che tale fattore è più grande del miglior risultato noto di non-approssimabilità per Max 3Sat.

Contributo innovativo. *Usiamo una nuova caratterizzazione dei grafi risultanti dalla riduzione di [46]. Era opinione corrente che tali grafi non fossere sparsi perché contenevano grandi cricche. Dimostriamo che il punto essenziale è che in effetti contengono grandi* sottografi bipartiti completi*, e che possono essere proprio caratterizzati come* unione *di grafi bipartiti completi. La nostra riduzione sostituisce* epxander bipartiti di grado limitato *al posto di tali grafi bipartiti completi.*

Capitolo 2

Definizioni preliminari

Symbol, *n. Something that is supposed to typify or stand for something else. (...) We cannot stop making them, but we can give them a name to conceal our helplessness.*

(A. Bierce [24])

Assumiamo che il lettore sia familiare con i concetti di base della teoria della complessità computazionale. Per le definizione delle classe di complessità usate in queste tesi rinviamo il lettore ad uno dei libri sull'argomento (ad esempio, [55, 14, 26, 95]). In questo capitolo richiameremo le definizioni di alcune riducibilità approximation-preserving, alcuni fatti su dimostrazioni verificabili probabilisticamente e query complexity, e definiremo i problemi che sono menzionati nel corso della tesi. Per cominciare, diamo le definizioni di problema di ottimizzazione e di algoritmo di approssimazione

2.1 Ottimizzazione e approssimazione

In questa sezione, daremo alcune definizioni standard nel campo della teoria dell'ottimizzazione e dell'approssimazione. Per ogni stringa x, denotiamo con $|x|$ la lunghezza della sua codifica binaria[1].

Definizione 2.1 (Problema NPO) *Un* problema di ottimizzazione NP *è una quadrupla* $A = (I, \mathsf{sol}, \mathsf{m}, \mathrm{type})$ *tale che*

1. *I è l'insieme delle* istanze *di A ed è riconoscibile in tempo polinomiale.*

2. *Data una istanza x di I, $\mathsf{sol}(x)$ denota l'insieme delle* soluzioni ammissibili *per x. Tali soluzioni sono brevi, cioè esiste un polinomio p tale che, per ogni $y \in \mathsf{sol}(x)$, $|y| \leq p(|x|)$. Inoltre, per ogni x ed ogni y tale che $|y| \leq p(|x|)$, è possibile decidere in tempo polinomiale se $y \in \mathsf{sol}(x)$.*

3. *Data una istanza x ed una soluzione ammissibile y per x, $\mathsf{m}(x, y)$ denote la* misura *intera positiva di y (spesso chiamata il* valore, *o* costo *di y). La funzione m è calcolabile in tempo polinomiale ed è anche chiamata la funzione* obiettivo.

4. $\mathrm{type} \in \{\max, \min\}$.

[1] Talvolta, per un'istanza $|x|$ di un problema di ottimizzazione, $|x|$ non sarà definita come la lunghezza della sua codifica binaria, ma come un altro parametro di interesse. As esempio, se x è un grafo, allora $|x|$ sarà tipicamente il numero di nodi. Questi casi speciali saranno specificati di volta in volta.

L'obiettivo di un problema di ottimizzazione NP rispetto ad una istanza x è trovare una *soluzione ottima*, cioè una soluzione ammissibile y tale che $\mathsf{m}(x,y) = \text{type}\{\mathsf{m}(x,y') : y' \in \mathsf{sol}(x)\}$. Nel seguito, opt denoterà la funzione che mappa una istanz x nella misura di una soluzione ottima. Per ogni istanza x, denotiamo inoltre con $\mathsf{sol}^*(x)$ l'insieme delle soluzioni ottime per x.

La *classe* NPO è l'insieme di tutti i problemi di ottimizzazione NP, $\mathsf{Max\ NPO}$ è l'insieme dei problemi di massimizzazione in NPO e $\mathsf{Min\ NPO}$ è l'insieme dei problemi di minimizzazione in NPO.

Un problema NPO è detto essere *polinomialmente limitato* se un polinomio q esiste tale che, per ogni istanza x e per ogni soluzione y di x, $\mathsf{m}(x,y) \leq q(|x|)$. La *classe* $\mathsf{NPO\ PB}$ è l'insieme do tutti i problemi NPO polinomialmente limitati. $\mathsf{Max\ PB}$ è l'insieme di tutti i problemi di massimizzazione in $\mathsf{NPO\ PB}$ e $\mathsf{Min\ PB}$ è l'insieme di tutti i problemi di minimizzazione in $\mathsf{NPO\ PB}$.

Definizione 2.2 (Rapporto di prestazione) *Sia A un problema* NPO. *Data una instanza x ed una solution ammissibile y per x, definiamo il* rapporto di prestazione di y rispetto ad x *come*

$$R(x,y) = \max\left\{\frac{\mathsf{m}(x,y)}{\mathsf{opt}(x)}, \frac{\mathsf{opt}(x)}{\mathsf{m}(x,y)}\right\}$$

*e l'*errorre relaivo di y rispetto a x *come*

$$E(x,y) = \frac{|\mathsf{opt}(x) - \mathsf{m}(x,y)|}{\mathsf{opt}(x)}.$$

Il rapporto di prestazione (rispettivamente, errore relativo) è sempre un numero maggiore o uguale ad 1 (rispettivamente, 0) ed è tanto più vicino ad 1 (rispettivamente, 0) quanto più il valore della soluzione ammissibile è vicino al valore ottimo. È facile vedere che, per ogni istanza x e per ogni soluzione ammissibile y prt x,

$$\frac{R(x,y)-1}{R(x,y)} \leq E(x,y) \leq R(x,y) - 1.$$

Definizione 2.3 (Algoritmo di approssimazione) *Sia A un problema* NPO *e sia T un algoritmo che, per ogni istanza x di A tale che $\mathsf{sol}(x) \neq \emptyset$, restituisce una soluzione ammissibile $T(x)$ in tempo polinomiale. Data una funzione $r : \mathcal{Z}^+ \to [1,\infty)$, diremo che T è un* algoritmo $r(n)$-approssimato per *A se il rapporto di prestazione della soluzione ammissibile $T(x)$ rispetto ad x verifica la seguente disuguaglianza:*

$$R(x,T(x)) \leq r(|x|).$$

Definizione 2.4 ($F\mathsf{APX}$) *Data una classe di funzioni F, un problema* NPO *A appartiene alla* classe F-APX *se esuste un algoritmo T $r(n)$-approssimato per A, per una qualche funzione $r \in F$.*

In particolare denoteremo con APX, $\log\mathsf{APX}$, $\text{poly}\mathsf{APX}$, e $\exp\mathsf{APX}$ le classi F-APX con F uguale all'insieme $O(1)$, all'insieme $O(\log n)$, all'insieme $O(n^{O(1)})$, e all'insieme $O(2^{n^{O(1)}})$, rispettivamente. Si potrebbe obiettare che non c'è differenza tra NPO ed $\exp\mathsf{APX}$, dato che il limite polinomiale sul tempo di calcolo della funzione obiettivo implica che ogni problema NPO è $h2^{n^k}$-approssimabile per qualche costante h e k. Questa intuizione è peroò falsa, dal momento che esistono problemi NPO per i quali è difficile anche solamente trovare una soluzione ammissibile. Vedremo esempi di tali problemi nella Sezione 3.2.1 (per esempio MAX ONES SAT). Per ogni classe $\mathcal{C} \in \{\mathsf{APX}, \log\mathsf{APX}, \text{poly}\mathsf{APX}\}$, denotiamo con $\mathcal{C}$ PB la classe $\mathcal{C} \cap \mathsf{NPO\ PB}$

Definizione 2.5 (APTAS) *Un problema $A \in$ NPO appartiene alla* classe APTAS *se esistono un algoritmo T ed una costante k tali che, per ogni $x \in I_A$ e per ogni $r > 1$, $R(x, T(x,r)) \leq r + k/\mathsf{opt}(x)$, e il tempo di calcolo di $T(x,r)$ è polinomiale in $|x|$.*

Definizione 2.6 (PTAS) *Un problema $A \in$ NPO appartiene alla* classe PTAS *se esiste un algoritmo T tale che, per ogni razionale fissato $r > 1$, $T(\cdot, r)$ è un algoritmo r-approssimato per A.*

Definizione 2.7 (FPTAS) *Un problema $A \in$ NPO appartiene alla* classe FPTAS *se esiste un algoritmo T tale che, per ogni $r > 1$ fissato, $T(\cdot, r)$ è un algoritmo r-approssimato per A il cui tempo di calcolo è polinomiale in $|x|$ e in $1/(r-1)$.*

Chiaramente, valgono le seguenti inclusioni:

$$\mathsf{FPTAS} \subseteq \mathsf{PTAS} \subseteq \mathsf{APTAS} \subseteq \mathsf{APX} \subseteq \text{log-}\mathsf{APX} \subseteq \text{poly-}\mathsf{APX} \subseteq \text{exp-}\mathsf{APX} \subseteq \mathsf{NPO}.$$

È anche facile vedere che tutte queste inclusioni sono strette se e solo se $\mathsf{P} \neq \mathsf{NP}$.

Facciamo riferimento a [94] (vedi anche [81]) per una definizione formale di Max NP e Max SNP. Daremo nel seguito una definizione informale ma equivalente. Diremo che un problema A è un *sottoproblema* di B se $I_A \subseteq I_B$, e per ogni $x \in I_A$ si da il caso che $\mathsf{sol}_A(x) = \mathsf{sol}_B(x)$ e $\mathsf{m}_A(x, \cdot) = \mathsf{m}_B(x, \cdot)$.

Definizione 2.8 (Max SNP) *Un problema* NPO *appartiene alla classe* Max SNP *se esiste una costante k take che A può essere espresso come un sottoproblema di* MAX *k-CSP.*

Definizione 2.9 (Max NP) *Un problema* NPO *è nella classe* Max NP *se una costante B esiste tale che A può essere espresso come un sottoproblema di* MAX *GSAT-B.*

Teorema 2.10 ([94]) Max SNP $\subset$ Max NP $\subset$ APX PB.

2.2 Riducibilità approximation-preserving

In generale, in teoria della complessità, una *riduzione* da un problema A ad un problema B specifica una qualche procedura per risolvere A attraverso un algoritmo che risolve B. Nel contesto dell'approssimazione, una riduzione dovrebbe garantire che una solutione approssimata per B può essere usata per ottenere una soluzione approssimata per A. A questo scopo, è necessaria una funzione f che mappi un'istanza x di A in un'istanza x' di B, e anche un'altra funzione g che mappi una soluzione approssimata y' per x' in una soluzione approssimata y per x (vedi la Figura 2.1). In effetti, molte delle riduzioni note tra problemi di ottimizzazione definiscono esplicitamente tale corrispondenza tra le soluzioni.

Formalizziamo questa intuizione introducendo la nozione di *schema di riduzione*.

Definizione 2.11 (Schema di riduzione) *Siano A e B be due problemi* NPO*, uno* schema di riduzione *(f, g) tra loro è una coppia ordinata di funzioni calcolabili in tempo polinomiale tali che le seguenti proprietà valgano.*

1. *Per ogni $x \in I_A$, $f(x) \in I_B$.*
2. *Per ogni $x \in I_A$, se $\mathsf{sol}(x) \neq \emptyset$ allora anche $\mathsf{sol}(f(x)) \neq \emptyset$.*

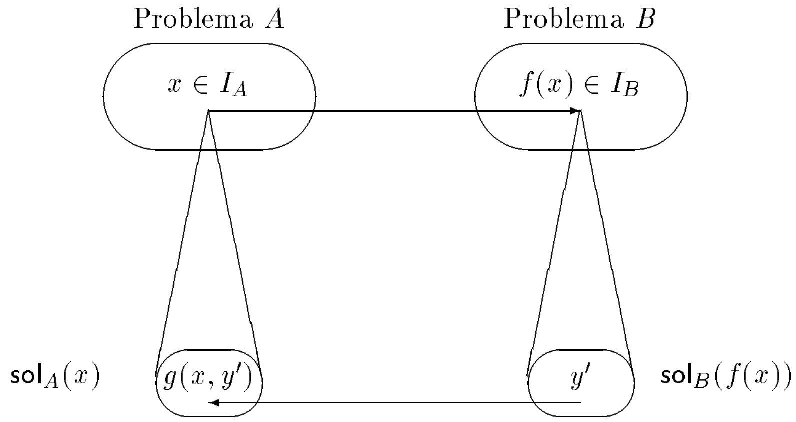


Figura 2.1: Riduzione tra due problemi di ottimizzazione.

3. *Per ogni* $y' \in \mathsf{sol}(f(x))$*,* $g(x, y') \in \mathsf{sol}(x)$*.*

La seconda condizione nella definizione precedente potrebbe sembrare poco naturale. La sua giustificazione è la seguente. Assumiamo che B sia un problema tale una istanza $x_{\mathrm{void}} \in I_B$ esista per la quale $\mathsf{sol}(x_{\mathrm{void}}) = \emptyset$ (per esempio, possiamo prendere $B =$ Linear Programming). Allora, senza la condizione 2, per ogni $A \in \mathsf{NPO}$, (f, g) avremmo uno schema di riduzione tra A e B purché $f(x) = x_{\mathrm{void}}$ per ogni $x \in I_A$. In effetti, in tal caso, la condizione 3 sarebbe sempre banalmente verificata. Le riducibilità approximation-preserving sono definite rafforzando la condizione 3: senza la condizione 2 ci sarebbero riducibilità banali tra ogni problema NPO e Linear Programming, così che Linear Programming (un problema risulubile in modo ottimo in tempo polinomiale) sarebbe NPO-completo.

Le riducibilità approximation-preserving sono definite imponendo qualche relazione tra il rapporto di prestazione di y' e quello di $g(x, y')$. La nozione più forte di riducibilità approximation-preserving è la S-riducibilità introdotta da Orponen e Manilla [92].

Definizione 2.12 (S-reducibilità) *Siano A e B due problemi* NPO*. Diremo che A è* S-riducibile *a B se esiste uno schema di riduzione (f, g) tra A e B tale che per ogni $x \in I_A$, e ogni $y' \in \mathsf{sol}(f(x))$, si abbia $E(x, g(x, y')) \leq E(f(x), y')$.*

Alcune riduzioni apparse nella letteratura sono S-riduzioni, come ad esempio la riduzione standard tra Max Clique e Max Independent Set. Nella Sezione 4.6.1 vedremo una S-riduzione tra Min Vertex Cover e Min Sat. È facile verificare che la S-riducibilità è transitiva e che preserva l'appartenenza a PTAS e APX.

La L-riducibilità introdotta da Papadimitriou e Yannakakis [94] è spesso la riducibilità più pratica da usare per dimostrare che un problema è almeno più difficile da approssimare di un altro.

Definizione 2.13 (L-riducibilità) *Siano A e B due problemi* NPO*. Diremo che A è* L-riducibile *a B, in simboli $A \leq_{\mathrm{L}} B$, se esiste uno schema di riduzione (f, g) tra A e B e due costanti positive α e β tali che:*

1. *Per ogni* $x \in I_A$*,* $\mathsf{opt}_B(f(x)) \leq \alpha \mathsf{opt}_A(x)$*.*

2. *Per ogni $x \in I_A$ e per ogni $y \in \mathsf{sol}_B(f(x))$,*

$$|\mathsf{opt}_A(x) - \mathsf{m}_A(x, g(x, y))| \leq \beta |\mathsf{opt}_B(f(x)) - \mathsf{m}_B(f(x), y)|$$

Chiameremo la quadrupla (f, g, α, β) una L-riduzione *da A a B.*

Chiaramente, la L-riducibilità conserva l'appartenenza a PTAS. Infatti, se (f, g, α, β) è una L-riduzione da A a B allora, per ogni $x \in I_A$ e per ogni $y \in sol_B(f(x))$, abbiamo che

$$E_A(x, g(x, y)) \leq \alpha\beta E_B(f(x), y),$$

così che se $B \in \mathsf{PTAS}$ allora anche $A \in \mathsf{PTAS}$ [94]. La disuguaglianza precedente implica anche che se A è un problema di minimizzazione ed esiste un algoritmo r-approssimato per B, allora esiste anche un algoritmo $(1 + \alpha\beta(r-1))$-approssimato. La L-riducibilità conserva l'approssimazione in un modo strano, imponendo vincoli separati sugli ottimi e sugli errori additivi nei due problemi. La E-riducibilità, introdatta da Khanna et al. [80], invece, impone una relazione lineare direttamente tra i rapporti di prestazione.

Definizione 2.14 (E-riducibilità) *Siano A e B due problemi* NPO. *Diremo che A è* E-riducibile *a B, in simboli $A \leq_{\mathrm{E}} B$, se esiste uno schema di riduzione (f, g) tra A e B eyba costante positiva α tale che per ogni $x \in I_A$ e per ogni $y \in sol_B(f(x))$,*

$$R_A(x, g(x, y)) \leq 1 + \alpha(R_B(f(x), y) - 1).$$

Chiameremo la tripla (f, g, α) The triple (f, g, α) is said to be an E-riduzione *da A a B.*

Si osservi che, per ogni funzione r, una E-riduzione mappa soluzioni $r(n)$-approssimate in soluzioni $(1 + \alpha(r(n^h) - 1))$-approssimate, dove h è una costante che dipende solo dalla riduzione. Dunque, la E-riducibilità non solo conserva l'appartenenza in PTAS, ma anche in expAPX, polyAPX, logAPX, e APX.

Altre riducibilità apparse nella letteratura sono la P-riducibilità e la A-riducibilità definite da Crescenzi e Panconesi [94] (simili riducibilità erano state anche definite in [92]).

Definizione 2.15 (P-riducibilità) *Siano A e B due problemi* NPO. *Diremo che A è* P-riducibile *a B, in simboli $A \leq_{\mathrm{P}} B$, se esiste uno schema di riduzione (f, g) tra A e B e una funzione calcolabile $c : (1, \infty) \cap Q^+ \to (1, \infty) \cap Q^+$ tali che, per ogni $x \in I_A$, per ogni $y \in sol_B(f(x))$, per ogni razionale $r > 1$,*

$$R_B(f(x), y) \leq c(r) \text{ implica } R_A(x, g(x, y)) \leq r \;.$$

Chiameremo la tripla (f, g, c) una P-riduzione *da A a B.*

Si noti che una E-riduzione è anche una P-riduzione con $c(r) = 1 + (r-1)/\alpha$, mentre una L-riduzione è anche una P-riduzione con $c(r) = 1 + \frac{r-1}{\alpha\beta r}$.

Definizione 2.16 (A-riducibilità) *Siano A e B due problemi* NPO. *Diremo che A è* A-riducibile *a B, in simboli $A \leq_{\mathrm{A}} B$, se esiste uno schema di riduzione (f, g) tra A e B e una funzione calcolabile $c : (1, \infty) \cap Q^+ \to (1, \infty) \cap Q^+$ tali che, per ogni $x \in I_A$, per ogni $y \in sol_B(f(x))$, per ogni razionale $r > 1$,*

$$R_A(x, g(x, y)) \leq c(r) \text{ implies } R_B(f(x), y) \leq r \;.$$

Chiameremo la tripla (f, g, c) una A-riduzione *da A a B.*

Il risultato seguente sarà usato nel prosimo capitolo.

Teorema 2.17 ([80]) *Sia $\mathcal{C} \in \{\mathsf{APX}, \log\mathsf{APX}, \mathrm{poly}\mathsf{APX}, \exp\mathsf{APX}\}$, sia A un problema di minimizzazione in $\mathcal{C}$. Allora esiste un problema di massimizzazione $B \in \mathcal{C}$ tale che $A \leq_{\mathrm{E}} B$ e $A \leq_{\mathrm{L}} B$.*

Teorema 2.18 ([94]) Max 3Sat *è* Max SNP*-completo rispetto alla L-riducibilità.*

2.3 PCP e la complessità dell'approssimazione

Nel seguito richiameremo le principali definizioni e alcuni risultati della teoria delle dimostrazioni verificabili probabilisticamente. La nostra esposizione segue [19].

Un *promise problem* è una coppia (Y, N) di insiemi disgiunti di stringhe su un certo alfabeto fissato Σ (possiamo assumere senza perdere in generalità che $\Sigma = \{0, 1\}$). Un algoritmo *risolve* un promise problem (Y, N) se per ogni stringa di input $x \in Y$ l'algoritmo accetta e se per ogni stringa di input $x \in N$ l'algoritmo rigetta. Il comportamente dell'algoritmo può essere arbitrario quando riceve una strina di input presa da $\Sigma^* - (Y \cup N)$. Un linguaggio L può essere visto come il promise problem $(L, \Sigma^* - L)$.

Un *verificatore* è una macchina di Turing V probabilistica con oracolo che funziona in tempo polinomiale. Durante la sua computazione, V usa random bit, legge il suo input, e ha accesso di tipo oracolo ad una stringa π chiamata *prova*. In particolare, sia a la sequenza di risposte di oracolo ricevute da V nel corso della sua computazione con input x e stringa random R. Se V accetta in questa particolare circostanza, allora diremo che (x, R, a) è una configurazione accettante per V. Sia ora x un input e π una prova. Denotiamo con $\mathbf{ACC}[V^\pi(x)]$ la probabilità, rispetto ai suoi random bit, che V accetti x usando π come oracolo. Denotiamo anche con $\mathbf{ACC}[V(x)]$ il massimo di $\mathbf{ACC}[V^\pi(x)]$ su tutte le prove π.

Numerosi parametri determinano l'efficienza della verifica della dimostrazione.

Definizione 2.19 (Parametri PCP) *Sia (Y, N) un promise problem, e sia V un verificatore per (Y, N). Diremo che:*

- V usa $r(n)$ random bit *(dove $r : \mathcal{Z}^+ \to \mathcal{Z}^+$ è una funzione intera) se per ogni input x e per ogni prova π, V accede al più a $r(|x|)$ random bit;*
- V ha complessità di bit letti q *(dove q è un intero) se per ogni input x, ogni stringa random R, e ogni prova π, V legge al più q bit da π;*
- V ha complessità di free bit f *(dove f è un reale) se per ogni input x e ogni stringa random R, ci sono al più 2^f insiemi di risposte a tali che (x, R, a) è una configurazione accettante per V.*
- V ha soundness s *(dove $s \in [0, 1]$ è un reale) se, per ogni $x \in N$, $\mathbf{ACC}[V(x)] \leq s$;*
- V ha completezza c *(dove $c \in [0, 1]$ è un reale) se, per ogni $x \in Y$, $\mathbf{ACC}[V(x)] \geq c$.*

Notiamo che il numero di free bit non può eccedere il numero di bit letti.

Definizione 2.20 (Classi PCP) *Sia (Y, N) in promise problem, siano $0 < s < c \leq 1$ costanti qualsiasi, sia $f > 0$, q un intero positivo e $r : \mathcal{Z}^+ \to \mathcal{Z}^+$. Diremo che $(Y, N) \in \mathsf{FPCP}_{c,s}[r, f, q]$ se esiste un verificatore V per (Y, N) che usi $O(r(n))$ bit random, abbia complessità di bit letti q, complessità di free bit f, soundness s e completezza c. $\mathsf{PCP}_{c,s}[r, q]$ è l'unione di $\mathsf{FPCP}_{c,s}[r, f, q]$ per tutti gli $f \leq q$.*

Usando la notazione precedente (introdotta in [19] e leggermente differente da quella di [6]), possiamo formulare il teorema PCP nel modo seguente.

Teorema 2.21 (Teorema PCP [6, 5]) *Esiste una costante q tale che*

$$\mathsf{NP} \subseteq \mathsf{PCP}_{1,1/2}[\log, q] \ .$$

Talvolta useremo anche la notazione abbreviata $\mathsf{PCP}(r(n), q(n)) \stackrel{\text{def}}{=} \mathsf{PCP}_{2/3,1/3}[r(n), O(q(n))]$. Si noti che le costanti 2/3 e 1/3 sono state scelte in modo arbitrario, cioè per ogni c e s fissati, con $0 < s < c < 1$ si ha $\mathsf{PCP}(r(n), q(n)) = \mathsf{PCP}_{c,s}[r(n), O(q(n))]$.

Il seguente teorema mostra che l'esistenza di verificatori efficienti per ogni problema NP implica un risultato di non-approssimabilità per Min Vertex Cover.

Teorema 2.22 (Non-approssimabilità di Min Vertex Cover [46, 19]) *Assumiamo che* $\mathsf{NP} \subseteq \mathsf{FPCP}_{c,s}[\log, f, q]$. *Allora, per ogni $\epsilon > 0$, è NP-hard approssimare* Min Vertex Cover *entro*

$$1 + \frac{c - s}{2^f - c} - \epsilon \ .$$

Dimostrazione: Sia ϕ una istanza del problema Sat, e consideriamo il comportamento del verificatore (la cui esistenza è data come ipotesi) con un generico input ϕ e dimostrazione π. Sia $\mathbf{r} = 2^{O(\log n)}$ il numero totale (polinomiale in n di possibili stringhe random a cui il verificatore può accedere. Per ognuna di tali sequenze R, vi sono al più 2^f diverse configurazioni accettanti (x, R, a). Diremo che due configurazioni (x, R, a) e (x, R', a') sono *consistenti* se esiste una dimostrazione π tale che a (rispettivamente, a') è l'insieme delle risposte ricevute durante la computazione $V^\pi(x, R)$ (rispettivamente, $V^\pi(x, R')$). Costruiamo il grafo G_ϕ che ha un nodo per ogni configurazione accettante (aggiungiamo eventualmente configurazioni fittizie per assicurarci che il numero totale di nodi sia esattamente $2^f\mathbf{r}$). In G_ϕ esiste un arco tra u e v se e solo se u e v non sono consistenti. È possibile dimostrare (vedi [46]) che cè un insieme indipendente in G_ϕ con almeno k nodi se e solo se cè una prova per ϕ che faccia accettare il verificatore con probabilità almeno k volte su $\mathbf{r}$ (cioè con prabailità almeno $k/\mathbf{r}$). Si osservi che un grafo G_ϕ con N nodi ha un insieme indipendente con k nodi se e solo se ha un vertex cover con $N - k$ nodi. Ne consegue che se ϕ è soddisfacibile, allora esiste un vertex cover in G_ϕ con al più $\mathbf{r}(2^f - c)$ nodi; altrimenti ogni vertex cover in G_ϕ avrà almeno $\mathbf{r}(2^f - s)$ nodi. Perciò, ogni fattore di approssimazione migliore di $(2^f - s)/(2^f - c)$ sarebbe sufficiente per decidere la soddisfacibilità di ϕ. □

Nel seguito, i grafi G_ϕ risultanti dalla costruzione precedentemente descritta saranno chiamati *grafi FGLSS*.

Al momento di scrivere questa tesi, la migliore costruzione PCP nota con pochi bit letti e pochi free bit è la seguente.

Teorema 2.23 (Håstad [63]) *Per ogni costante $\epsilon > 0$,* $\mathsf{NP} \subseteq \mathsf{FPCP}_{1-\epsilon,.5+\epsilon}[\log, 2, 3]$.

Di conseguenza, approssimare Min Vertex Cover entro 1.166 è NP-hard.

2.4 Query complexity

In questa sezione daremo alcune definizioni e risultati su classi di complessità definite in termini di *domande limitate* ad un *oracolo* NP.

Definizione 2.24 *Sia $q : \mathcal{Z}^+ \to \mathcal{Z}^+$. Un linguaggio L appartiene alla* classe $\mathsf{P}^{\mathsf{NP}[q(n)]}$ *se e solo se è decidibile da una macchina di Turing polinomiale con oracolo che chiede al più $q(n)$ domande ad un oracolo* NP*, dove n è la dimensione dell'input. Inoltre, per una classe Q di funzioni, definiamo* $\mathsf{P}^{\mathsf{NP}[Q]} \stackrel{\text{def}}{=} \bigcup_{q \in Q} \mathsf{P}^{\mathsf{NP}[q(n)]}$.

La *classe* QH (Query Hierarchy) è uguale all'unione $\bigcup_{q \geq 1} \mathsf{P}^{\mathsf{NP}[q]}$. Definiamo inoltre $\mathsf{P}^{\mathsf{NP}} \stackrel{\text{def}}{=} \mathsf{P}^{\mathsf{NP}[n^{O(1)}]}$. Il seguente risultato di *separazione* per la calsse QH è dovuto a Kadin.

Teorema 2.25 ([72, 73]) *Se esiste una costante q tale che*

$$\mathsf{QH} = \mathsf{P}^{\mathsf{NP}[q]},$$

allora la gerarchia polinomiale collassa.

Wagner ha esteso il risultato precedente al caso di un numero non costante di domande.

Teorema 2.26 ([108]) *Sia $q(n) \leq \log n$ una funzione calcolabile in tempo polinomiale. Se* $\mathsf{P}^{\mathsf{NP}[q(n)]} \subseteq \mathsf{P}^{\mathsf{NP}[q(n)-1]}$ *allora la gerarchia polinomiale collassa.*

È anche possibile (e talvolta utile) definire classi di *funzioni* che siano calcolabili usando un numero limitato di domande.

Definizione 2.27 *Una funzione $f : \mathcal{Z}^+ \to \mathcal{Z}^+$ appartiene alla* classe $\mathsf{PF}^{\mathsf{NP}[q(n)]}$ *se e solo se è calcolabile da una macchina di Turing polinomiale con oracolo che chiede al più $q(n)$ domande ad un oracolo* NP*, dove n è la dimensione dell'input. Inoltre, se A è una classe di funzioni , allora* $\mathsf{PF}^{\mathsf{NP}[Q]} = \bigcup_{q \in Q} \mathsf{PF}^{\mathsf{NP}[q(n)]}$.

2.5 Definizioni dei problemi

In questa sezione daremo le definizioni di tutti i problemi di ottimizzazione che sono menzionati in questa tesi. Seguendo il formato del compendio di Crescenzi e Kann [33], definiremo per ogni problema l'insieme delle istanze, l'insieme delle soluzioni ammissibili per una istanza data e la funzione obiettivo. Non specifichiamo se il problema è di massimizzazione o di minimizzazione dato che è evidente dal nome.

2.5.1 Soddisfazione di vincoli

Definizione 2.28 *Una* funzione di vincolo *k-aria è una funzione booleana $f : \{0,1\}^k \to \{0,1\}$.*

Quando è applicata a variabili $x_1, \ldots, x_k$ (vedi le definizioni seguenti) la funzione f impone il vincolo $f(x_1, \ldots, x_k) = 1$.

Definizione 2.29 *Una* famiglia di vincoli *$\mathcal{F}$ è una collezione finita di funzioni di vincolo. L'*arità *di $\mathcal{F}$ è la massima arità delle funzioni in $\mathcal{F}$.*

Definizione 2.30 *Un* vincolo *C su un insieme di variabili $x_1, \ldots, x_n$ è una coppia $C = (f, (i_1, \ldots, i_k))$ dove $f : \{0,1\}^k \to \{0,1\}$ è una funzione di vincolo e $i_j \in [n]$ for $j \in [k]$. Si dice che la variabile x_j* occorre *(o* ha una occorrenza*) in C se $j \in \{i_1, \ldots, i_k\}$. Si dice che il vincolo C è* soddisfatto *da un assegnamento $\vec{a} = a_1, \ldots, a_n$ a $x_1, \ldots, x_n$ se $C(a_1, \ldots, a_n) \stackrel{\text{def}}{=} f(a_{i_1}, \ldots, a_{i_k}) = 1$. Diremo anche che il vincolo C è* da *$\mathcal{F}$ se $f \in F$.*

Problema 1 (Max Weight $\mathcal{F}$, Max $\mathcal{F}$, Max Simple $\mathcal{F}$) *Per ogni famiglia di funzioni $\mathcal{F}$, il* problema di soddisfacimento di vincoli Max Weight $\mathcal{F}$ *è il problema di ottimizzazione definito nel modo seguente:*

Istanza: *Una collezione $\phi = \{C_1, \ldots, C_m\}$ di vincoli su un insieme di variabili $X = \{x_1, \ldots, x_n\}$ e i pesi non negativi associati $w_1, \ldots, w_m$.*

Soluzione: *Un assegnamento di valori di verità τ per le variabili in X.*

Misura: *Peso totale dei vincoli soddisfatti dall'assegnamento di valori di verità.*

Max Simple $\mathcal{F}$ *è la restrizione di* Max Weight $\mathcal{F}$ *dove tutti i pesi sono uguali a 1.* Max $\mathcal{F}$ *è la variazione di* Max Simple $\mathcal{F}$ *dove si ammette che ϕ sia un multinsieme di clausole. Equivalentemente,* Max $\mathcal{F}$ *è la variazione di* Max Weight $\mathcal{F}$ *in cui i pesi sono rappresentati in unario.*

Diamo ora una lista di tutti i vincoli e di tutte le famiglie di vincoli che saranno usati in questa tesi.

Definizione 2.31 *Definiamo le seguenti famiglie di vincoli.*

- Test di parità *è la famiglia di vincoli* $\mathrm{PC} = \{\mathrm{PC}_0, \mathrm{PC}_1\}$, *dove, fper* $i \in \{0,1\}$, PC_i *è definito nel modo seguente:*

$$\mathrm{PC}_i(a_1, a_2, a_3) = \begin{cases} 1 & \textit{if } a_1 \oplus a_2 \oplus a_3 = i \\ 0 & \textit{altrimenti.} \end{cases}$$

- Test di rispetto della base monomiale *è la famiglia di vincoli* $\mathrm{RMBC} = \{\mathrm{RMBC}_{ij} : i, j \in \{0,1\}\}$, *dove* $\mathrm{RMBC} : \{0,1\}^4 \to \{0,1\}$ *with*

$$\mathrm{RMBC}_{ij}(a_1, a_2, a_3, a_4) = \begin{cases} 1 & \textit{se } a_1 = 0 \textit{ e } a_2 = a_3 \oplus i \\ 1 & \textit{se } a_1 = 1 \textit{ e } a_2 = a_4 \oplus j \\ 0 & \textit{altrimenti.} \end{cases}$$

RMBC_{00} *può essere pensato come il test* $(a_3, a_4)[a_1] \stackrel{?}{=} a_2$, RMBC_{01} *come il test* $(a_3, \neg a_4)[a_1] \stackrel{?}{=} a_2$, RMBC_{10} *come il test* $(\neg a_3, a_4)[a_1] \stackrel{?}{=} a_2$, *e* RMBC_{11} *come test* $(a_3, a_4)[a_1] \stackrel{?}{=} \neg a_2$.

- *Per ogni $k \geq 1$,* Esattamente-k-SAT *è la famiglia di vincoli* $\mathrm{E}k\mathrm{SAT} = \{f : \{0,1\}^k \to \{0,1\} : |\{\vec{a} : f(\vec{a}) = 0\}| = 1\}$, *cioè l'insieme dei vincoli* disgiuntivi *k-ari.*

- *Per ogni $k \geq 1$,* k-SAT *è la famiglia di vincoli* $k\mathrm{SAT} = \bigcup_{l \in [k]} \mathrm{E}l\mathrm{SAT}$.

- SAT *è la famiglia di vincoli* $\mathrm{SAT} = \bigcup_{k \geq 1} \mathrm{E}k\mathrm{SAT}$

- *Per ogni $B \geq 1$,* SAT-B Generalizzato *è la famiglia di vincoli* GSAT-B *che consiste di tutte le funzioni booleane che ammettono una rappresentazione DNF in cui tutte le congiunzioni hanno al più B termini.*

- *Per ogni $k \geq 1$* k-congiunzioni SAT *è la famiglia di vincoli* $k\mathrm{ConjSAT} = \{f : \{0,1\}^h \to \{0,1\} : |\{\vec{a} : f(\vec{a}) = 1\}| = 1 \wedge h \leq k\}$, *cioè l'insieme dei vincoli* congiuntivi *con al più k letterali.*

- CUT *è la funzione di vincolo* $\text{Cut} : \{0,1\}^2 \to \{0,1\}$ *con* $\text{Cut}(a_1, a_2) = a_1 \oplus a_2$.
- DICUT*è la funzione di vincolo* $\text{DiCut} : \{0,1\}^2 \to \{0,1\}$ *con* $\text{DiCut}(a_1, a_2) = \neg a_1 \wedge a_2$.
- *Per ogni* $k \geq 1$, k-CSP *è la famiglia di vincoli che consiste di tutte le funzione booleane di arità al più* k*, cioè.* $k\text{-CSP} = \{f : \{0,1\}^h \to \{0,1\} \ : \ h \leq k\}$.

La definizione di MAX WEIGHT CUT, MAX CUT, MAX WEIGHT DICUT, e MAX DICUT come problemi di soddisfacimento di vincoli è equivalente alla definizione standard come problemi su grafi.

Per ogni famiglia di vincoli $\mathcal{F}$, oltre ai problemi MAX WEIGHT $\mathcal{F}$, MAX $\mathcal{F}$ e MAX SIMPLE $\mathcal{F}$, definiamo anche i cosiddetti problemi max ones e min ones.

Problema 2 (MAX ONES $\mathcal{F}$ **e** MIN ONES $\mathcal{F}$) *Per ogni famiglia di vincoli* $\mathcal{F}$*, il problema* MAX ONES $\mathcal{F}$ *(rispettivamente, il problema* MIN ONES $\mathcal{F}$*) è definito nel modo seguente.*

ISTANZA: *Una collezione* $\phi = \{C_1, \ldots, C_m\}$ *di vincoli da* $\mathcal{F}$ *sull'insieme di variabili* $X = \{x_1, \ldots, x_n\}$ *e pesi associati non negativi* $w_1, \ldots, w_n$.

SOLUZIONE: *Un assegamento di valori di verità* τ *per le variabili in* X *che soddisfi tutti i vincoli in* ϕ.

MISURA: *Peso totale delle variabili assegnate a* true *da* τ.

2.5.2 Ricoprimento e impacchetamento

Problema 3 (MIN BIN PACKING)

ISTANZA: *Insieme finito* U *di oggetti, e una dimensione* $s(u) \in Q \cap (0,1]$ *per ogni* $u \in U$.

SOLUZIONE: *Una partizione di* U *in insiemi disgiunti* $U_1, U_2, \ldots, U_m$ *tale che la somma delle dimensioni degli oggetti in ogni* U_i *sia al più 1.*

MISURA: *Il numero* m *di insiemi disgiunti.*

Problema 4 (MIN ORDERED BIN PACKING)

ISTANZA: *Insieme finito* U *di oggetti, una dimensione* $s(u) \in Q \cap (0,1]$ *per ogni* $u \in U$*, e un ordine parziale* $\preceq$ *on* U.

SOLUZIONE: *Una partizione di* U *in insiemi disgiunti* $U_1, U_2, \ldots, U_m$ *tale che la somma delle dimensioni degli oggetti in ogni* U_i *sia al più 1 e se* $u \in U_i$ *e* $u' \in U_j$ *con* $u \preceq u'$*, allora* $i \leq j$.

MISURA: *Il numero* m *di insiemi disgiunti.*

Problema 5 (MIN DEGREE SPANNING TREE)

ISTANZA: *Grafo* $G = (V, E)$.

SOLUZIONE: *Un albero di copertura per* G.

MISURA: *Il grado massimo dell'albero di copertura.*

Problema 6 (MIN WEIGHT VERTEX COVER)

ISTANZA: *Grafo* $G = (V, E)$ *e pesi* $\{w(u)\}_{u \in V}$.

SOLUZIONE: *Un vertex cover per* G*, cioè un insieme* $C \subseteq V$ *tale che* C *contiene almeno un estremo di ogni arco di* E.

MISURA: *Peso totale dei vertici in* C*, cioè* $\sum_{u \in C} w(u)$.

Il problema MIN VERTEX COVER è la restrizione del problema MIN WEIGHT VERTEX COVER dove tutti i pesi sono 1. Per ogni costant $B \geq 1$, MIN VERTEX COVER-B è la restrizione del problema MIN VERTEX COVER a grafi di grado massimo B.

Problema 7 (MIN SET COVER)

ISTANZA: *Famiglia di sottoinsiemi* $S_1, \ldots, S_m$ *di un insieme* U *tale che* $\bigcup_{i=1}^m S_i = U$.

SOLUZIONE: *Un ricoprimento di* U*, cioè una sottofamiglia* $C = \{S_{i_1}, \ldots, S_{i_k}\} \subseteq \{S_1, \ldots, S_m\}$ *tale che* $\bigcup_{j=1}^k S_{i_j} = U$.

MISURA: *La cardinalità del ricoprimento, cioè* k.

Problema 8 (MIN HITTING SET)

ISTANZA: *Famiglia di sottoinsiemi* $S_1, \ldots, S_m$ *di un insieme* U.

SOLUZIONE: *Un sottoinsieme* $H \subseteq U$ *tale che per ogni* $i = 1, \ldots, m$ $H \cap S_i \neq \emptyset$.

MISURA: *La cardinalità di* H.

Problema 9 (MIN DOMINATING SET)

ISTANZA: *Un grafo* $G = (V, E)$.

SOLUZIONE: *Un sottoinsieme* $D \subseteq V$ *tale che per ogni* $u \in V - D$ *esiste un* $v \in D$ *tale che* u *e* v *sono adiacenti in* G.

MISURA: *La cardinalità di* D.

2.5.3 Cricca e Colorazione

Problema 10 (MAX WEIGHT CLIQUE)

ISTANZA: *Grafo* $G = (V, E)$ *e pesi* $\{w(u)\}_{u \in V}$.

SOLUZIONE: *Una cricca in* G*, cioè un sottoinsieme* $V' \subseteq V$ *tale che ogni coppia di vertici in* V' *è congiunta da un arco in* E.

MISURA: *Peso totale della cricca, cioè* $\sum_{u \in V'} w(u)$.

Il problema MAX CLIQUE è la restrizione del problema MAX WEIGHT CLIQUE dove tutti i pesi sono uguali a 1.

Problema 11 (MIN EDGE COLORING)

ISTANZA: *Grafo* $G = (V, E)$.

SOLUZIONE: *Una colorazione di* E*, cioè una partizione di* E *in insiemi disgiunti* $E_1, E_2, \ldots, E_k$ *tale che, per* $1 \leq i \leq k$*, nessuna coppia di archi in* E_i *condivida un estremo comune in* G.

MISURA: *Cardinalità della colorazione, cioè numero* k *di insiemi disgiunti.*

Problema 12 (MIN NODE COLORING)

ISTANZA: *Grsfo* $G = (V, E)$.

SOLUZIONE: *Una colorazione di* V*, cioè una partizione di* V *in insiemi disgiunti* $V_1, V_2, \ldots, V_k$ *tale che, per* $1 \leq i \leq k$*, nessuna coppia di vertici in* V_i *è adiacente in* G.

MISURA: *Cardinalità della colorazione, cioè numero* k *di insiemi disgiunti.*

2.5.4 Programmazione matematica

Problema 13 (LINEAR PROGRAMMING)

ISTANZA: *Matrice* $m \times n$ *di razionali* A*, vettore* b *di razionali di dimensione* m*, vettore* c *di razionali di dimensione* n.

SOLUZIONE: *Un vettore di razionali* x *di dimensione* n *tale che* $Ax \geq b$.

MISURA: $1 + \sum_{i=1}^{n} c_i x_i$.

Problema 14 (MAX PB $0-1$ PROGRAMMING **e** MIN PB $0-1$ PROGRAMMING)

ISTANZA: *Matrice* $m \times n$ *di interi* A*, vettore intero* b *di dimensione* m*, vettore binario* c *di dimensione* n.

SOLUZIONE: *Un vettore* x *binario di dimensione* n *tale che* $Ax \geq b$.

MISURA: $1 + \sum_{i=1}^{n} c_i x_i$.

2.5.5 Rimozione di archi

Problema 15 (MIN FEEDBACK VERTEX SET)

ISTANZA: *Un grafo orientato* $G = (V, E)$.

SOLUZIONE: *Un insieme di veritici* $S \subseteq V$ *tale che* $V - S$ *induce un sottografo aciclico.*

MISURA: *Cardinalità di* S.

Problema 16 (MIN CLAUSE DELETION 2SAT)

ISTANZA: *Un insieme* $\phi = \{C_1, \ldots, C_m\}$ *di vincoli da* 2SAT *sull'insieme di variabili* X.

SOLUZIONE: *Un sottoinsieme* $S \subset \phi$ *du vincoli tali che* $\phi - S$ *è soddisfacibile.*

MISURA: *Cardinalità di S.*

Problema 17 (MIN EDGE DELETION BIPARTITE SUBGRAPH)

ISTANZA: *Un grafo $G = (V, E)$.*

SOLUZIONE: *Un insieme di archi $E' \subseteq E$ tale che $G' = (V, E - E')$ è bipartito.*

MISURA: *Cardinalità di E'.*

2.5.6 Altri problemi

Problema 18 (MIN STEINER TREE)

ISTANZA: *Un grafo completo $G = (V, E)$ con associati pesi $\{d(u, v)\}_{(u,v)\in E}$ e un insieme di vertici speciali $U \subseteq V$.*

SOLUZIONE: *Un insieme di vertici di Steiner $S \subseteq V - U$ tale che $S \cup U$ induce un sottografo connesso $G_{S,U}$ di G, e un albero di copertura di $G_{S,U}$.*

MISURA: *Costo dell'albero di copertura.*

Problema 19 (MIN LINEAR ARRANGMENT)

ISTANZA: *Un grafo $G = (V, E)$ con n vertici e associati pesi $\{w(u, v)\}_{(u,v)\in E}$.*

SOLUZIONE: *Una biiezione $\pi : V \to [n]$.*

MISURA: $\sum_{(u,v)\in E} w(u, v)|\pi(u) - \pi(v)|$.

Capitolo 3

La struttura delle classi di approssimazione

There is a theory which states that se ever anyone discovers exactly what the Universe is for e why it is here, it will istantly disappear e be replaced by something even more bizarre e inexplicable.
There is another which states that this has already happened

(D. Adams [1])

Il primo obiettivo di questo capitolo è definire una riduzione approximation-preserving che possa essere usati per il maggior numero possibile di classi di approssimazione e tale che tutte le riduzioni apparse finora nella letteratura continuino a valre. Nonostante la L-riduciblità sia stata la più usata finore, daremo una forte evidenza che non può essere usata per ottenere risultati di completezza in classi "definite computazionalmente" come APX, logAPX, e polyAPX. Infatti, da un lato mostreremo che la L-riducibilità è *troppo debole* e non preserva l'approssimabilità entro un fattore costante (a meno che $\mathsf{P} = \mathsf{NP} \cap co\text{-}\mathsf{NP}$); d'altra parte è *troppo restrittiva* in quanto non consente di ridurre problemi che sono facili da approssimare ad altri che sono molto più difficili (a meno che $\mathsf{P}^{\mathsf{NP}} = \mathsf{P}^{\mathsf{NP}[O(\log n)]}$).

Introduciamo un nuovo tipo di riduzione approximation-preserving che supera tali limitazioni e che consenti di dimostrare tutti i risultati di completezza in classi di approssimazione apparsi nella letteratura, più alcuni nuovi.

In particolare, dimostriamo l'esistenza di problemi naturali NPO-completi (in precedenza, erano noti solo problemi Max NPO-completi e problemi Min NPO-completi), di problemi naturali NPO PB-completi (di nuovo, superando la dicotomia tra massimizzazione e minimizzazione) di problemi naturali APX-completi (in precedenza erano solo problemi APX PB-completi e problemi APX-completi artificiali), e di problemi naturali logAPX-completi e polyAPX-completi (risultati precedenti erano confinati a logAPX PB e a polyAPX PB, rispettivamente).

Tutti i risultati di completezza menzionati in precedenza (ad eccezione di quelli di NPO- e NPO PB-completezza) usano il teorema PCP nella loro dimostrazione. Mostriamo che un risultato di completezza più semplice, ma non banale, può essere ottenuto con una riduzione diretta. In particolare, diamo una semplice dimostrazione della Max NP-completezza di Max 3Sat (nessun risulato di Max NP-completezza era noto in precedenza senza fare uso di tecniche PCP). La nostra riduzione fornisce una risposta soddisfacente alla questione di perché i problemi in Max NP e Max SNP siano ugualmente difficili da approssimare e ha interessanti applicazioni algoritmiche. Infatti, la utilizziamo per migliorare il tempo di calcolo di uno schema di approssimazione precedentemente noto per istanze "planari" di problemi di soddisfacibilità.

Un altro gruppo di risultati si riferisce a nuove connessioni tra le proprietà di approssimazione e la query complexity di problemi di ottimizzazione NP-hard. In alcuni articoli recenti, si è mostrato che la nozione di query complexity (cioè, il numero di domande ad un oracolo NP necessarie per risolvere un dato problema) è uno strumento molto utile per investigare la complessità dei problemi di approssimazione. Tali risultati precedenti includono limitazioni superiori ed inferiori al numero di domande che sono necessarie per approssimare certi problemi di ottimizzazione (come Max Sat e Max Clique): questi risultati riguardano la complessità di approssimare il valore di una soluzione ottima, e non la complessità di *trovare* soluzioni approssimate. In questo capitolo, invece, sarà considerata la complessità "costruttiva" dell'approssimazione, considerando linguaggi che possono essere riconosciuti da macchine di Turing polinomiali che hanno un oracolo che risolve il problema di approssimazione. Dimostreremo che la completezza in classi di approssimazione implica limiti inferiori alla query complexity. In effetti, riusciamo a dire di più: dimostriamo che i due approcci sono essenzialmente equivalenti in quanto diamo condizioni necessarie e sufficienti per completezza in classi di approssimaizone in termini di query complexity e di proprietà combinatorie. L'importanza di questi risultati è duplice: danno una nuova intuizione sulla struttura dei problemi completi in classi di approssimazione e riconciliano l'approccio basato su modelli di calcolo standard con l'approccio basato sul modello proposto in [27]. Come osservazione finale, i nostri risultati possono essere visti come estensione di un risultato di [80] che dava una condizione necessaria (ma non sufficiente) per la APX-completezza.

Come applicazioni dei nostri risultati di query complexity, mostreremo l'esistenza di problemi naturali APX-intermedi, risolveremo una questione aperta di Chang [28] sulla approssimabilità di Max Clique, e proveremo che il problema Min Set Cover non è self-improvable (tutti questi risultati sono dimostrati sotto l'assunzione che la gerarchia polinomiale non collassi).

3.1 Riducibilità

In questa sezione definiremo una nuova riducibilità approximation-preserving. Giustificheremo la nostra definizione enfatizzando gli svantaggi di quelle precedentemente note.

3.1.1 La L-riducibilità

Teorema 3.1 *I seguenti enunciati sono equivalenti:*

1. *Esistono due problemi* $A \in \mathsf{Max\ NPO}$ *e* $B \in \mathsf{Min\ NPO}$ *tali che* $A \notin \mathsf{APX}$, $B \in \mathsf{APX}$, *e* $A \leq_{\mathrm{L}} B$.
2. *Esistono due problemi* $A, B \in \mathsf{Max\ NPO}$ *tali che* $A \notin \mathsf{APX}$, $B \in \mathsf{APX}$, *e* $A \leq_{\mathrm{L}} B$.
3. *Esiste un insieme di formulea booleane riconoscibile in tempo polinomiale tale che nessun algoritmo polinomiale può calcolare, per ognuna di esse, un assegnamento che la soddisfi.*

Dimostrazione:

(1) $\Rightarrow$ (2). In questo caso è sufficiente L-ridurre B ad un problema di massimizzazione C in APX (vedi Teorema 2.17).

(2) $\Rightarrow$ (3). Assumiamo che per ogni insieme di formule booleane riconoscibile in tempo polinomiale esista un algoritmo polinomiale che calcoli, per ogni formula nell'insieme, un assegnamento

che la soddisfi. Supponiamo che (f, g, α, β) sia una L-riduzione da un problema di massimizzazione A ad un problema di massimizzazione B e che B sia r-approssimabile per qualche $r > 1$. Sia x una istanza di A e sia y una soluzione per $f(x)$ tale che $\mathsf{opt}_B(f(x))/\mathsf{m}_B(f(x), y) \leq r$. Per alleggerire la notazione, sia $\mathsf{opt}_A = \mathsf{opt}_A(x)$, $\mathsf{m}_A = \mathsf{m}_A(x, g(x, y))$, $\mathsf{opt}_B = \mathsf{opt}_B(f(x))$, e $\mathsf{m}_B = \mathsf{m}_B(f(x), y)$, $\mathsf{m}_x = \max\{\mathsf{m}_A, \mathsf{m}_B/\alpha\}$. Dato che $\mathsf{m}_A \leq \mathsf{opt}_A$ e $\mathsf{m}_B/\alpha \leq \mathsf{opt}_B/\alpha \leq \mathsf{opt}_A$, abbiamo che $\mathsf{m}_x \leq \mathsf{opt}_A$. Mostriamo adesso che $\mathsf{opt}_A/\mathsf{m}_x \leq 1 + \alpha\beta(r-1)$, cioè, m_x approssima il valore di opt_A. Sia $\gamma = \frac{\alpha r}{1+\alpha\beta(r-1)}$. Ci sono due casi da considerare.

1. $\mathsf{opt}_B \leq \gamma \mathsf{opt}_A$. Per definizione di L-riducibilità, $\mathsf{opt}_A - \mathsf{m}_A \leq \beta(\mathsf{opt}_B - \mathsf{m}_B)$. Dato che $\mathsf{opt}_B \leq \gamma \mathsf{opt}_A$ e $\mathsf{opt}_B/\mathsf{m}_B \leq r$, abbiamo che

$$\frac{\mathsf{opt}_A - \mathsf{m}_A}{\mathsf{opt}_A} \leq \gamma\beta \frac{\mathsf{opt}_B - \mathsf{m}_B}{\mathsf{opt}_B} \leq \gamma\beta(1 - 1/r).$$

   Dunque,

$$\frac{\mathsf{opt}_A}{\mathsf{m}_x} \leq \frac{\mathsf{opt}_A}{\mathsf{m}_A} \leq \frac{1}{1 - \gamma\beta\frac{r-1}{r}} = 1 + \alpha\beta(r-1)$$

   dove l'ultima uguaglianza è dovuta alla definizione di γ.

2. $\mathsf{opt}_B > \gamma \mathsf{opt}_A$. Abbiamo che

$$\begin{aligned}
\frac{\mathsf{opt}_A}{\mathsf{m}_x} &\leq \frac{\mathsf{opt}_A}{\mathsf{m}_B/\alpha} \\
&< \frac{\alpha(\mathsf{opt}_B/\gamma)}{\mathsf{m}_B} \text{ (since } \mathsf{opt}_A < \mathsf{opt}_B/\gamma) \\
&\leq \frac{\alpha(\mathsf{opt}_B/\gamma)}{(\mathsf{opt}_B/r)} \text{ (since } \mathsf{m}_B \geq \mathsf{opt}_B/r) \\
&= \frac{\alpha r}{\gamma} \\
&= 1 + \alpha\beta(r-1).
\end{aligned}$$

Consideriamo adesso il seguente algoritmo polinomiale non deterministico.

begin {input: $x \in I_A$}
  calcola m_x usando l'algoritmo r-approssimato per B e la L-riduzione da A a B;
  guess $y \in \mathsf{sol}_A(x)$;
  if $\mathsf{m}_A(x, y) \geq \mathsf{m}_x$ **then** accept **else** reject;
end;

Applicando la riduzione di Cook [32] all'algoritmo precedente, otteniamo che, per ogni $x \in I_A$, è possibile costruire in tempo polinomiale una formula soddisfacibile ϕ_x tale che ogni assegnamento che soddisfa ϕ_x codifica una soluzione per x la cui misura sia almeno m_x. Inoltre, l'insieme $\{\phi_x : x \in I_A\}$ è riconoscibile in tempo polinomiale. È allora possibile, per ipotesi, calcolare in tempo polinomiale un assegnamento che soddisfi ϕ_x e dunque trovare una soluzione approssimata per x.

(3) $\Rightarrow$ (1). Assumiamo che esista un insieme S di formule booleane soddisfacibili tali che nessun algoritmo polinomiale può clacolare, per ognuna di esse, un assegnamento che la soddisfi.

Consideriamo i seguenti due problemi NPO $A = (I_A, \mathsf{sol}_A, \mathsf{m}_A, \max)$ e $B = (I_B, \mathsf{sol}_B, \mathsf{m}_B, \min)$, dove $I_A = I_B = S$, $\mathsf{sol}_A(x) = \mathsf{sol}_B(x) = \{y : y$ è un assegnamento per le varaiabili di $x\}$,

$$\mathsf{m}_A(x,y) = \begin{cases} |x| & \text{if } y \text{ is a satisfying assignment for } x, \\ 1 & \text{otherwise,} \end{cases}$$

e

$$\mathsf{m}_B(x,y) = \begin{cases} |x| & \text{if } y \text{ is a satisfying assignment for } x, \\ 2|x| & \text{otherwise.} \end{cases}$$

Chiaramente, B è in APX, mentre se A è in APX allora esiste un algoritmo polinomiale che calcolo, per ogni formula in S, un assegnamento che la soddisfa, il che contraddice l'ipotesi. Inoltre, è facile vedere che A è L-riducibile a B usando $f \equiv \lambda x.x$, $g \equiv \lambda x \lambda y.y$, $\alpha = 1$, e $\beta = 1$. □

Si noti che in [86] viene dimostrato che il terzo enunciato nel teorema precedente vale se e solo se la

it is shown that the third statement of the γ-riducibilità è differente dalla riducibilità many-one. Inoltre, in [70] viene mostrato che quest'ultima ipotesi è in qualche modo intermedia tra $\mathsf{P} \neq \mathsf{NP} \cap co\text{-}\mathsf{NP}$ and $\mathsf{P} \neq \mathsf{NP}$.

Corollario 3.2 *La L-riducibilità non conserva l'appartenenza ad* APX *a meno che* $\mathsf{P} = \mathsf{NP} \cap \text{co-}\mathsf{NP}$.

3.1.2 La E-riducibilità

Dal momento che la E-riducibilità conserva l'appartenenza ad APX, abbiamo che esistono problemi NPO che sono L-riducibili uno all'altro ma che non sono E-riducibili. D'altra parte, il seguente risultato mostra che all'interno della classe APX la E-riducibilità è solo una generalizzazione della L-riducibilità.

Teorema 3.3 *Per ogni due problemi* NPO *A e B, se $A \leq_{\mathrm{L}} B$ e $A \in$* APX*, allora $A \leq_{\mathrm{E}} B$.*

Dimostrazione: Sia T un algoritmo r-approssimato per A e sia $(f_L, g_L, \alpha_L, \beta_L)$ una L-riduzione da A a B. Allora, per ogni $x \in I_A$ e per ogni $y \in \mathsf{sol}_B(f_L(x))$, $E_A(x, g_L(x,y)) \leq \alpha_L \beta_L E_B(f_L(x), y)$. Se A è un problema di minimizzazione, allora, per ogni $x \in I_A$ and per ogni $y \in \mathsf{sol}_B(f_L(x))$,

$$\begin{aligned} R_A(x, g_L(x,y)) &= 1 + E_A(x, g_L(x,y)) \\ &\leq 1 + \alpha_L \beta_L E_B(f_L(x), y) \\ &\leq 1 + \alpha_L \beta_L (R_B(f_L(x), y) - 1), \end{aligned}$$

e perciò $(f_L, g_L, \alpha_L \beta_L)$ è una E-riduzione da A a B. Altrimenti (vale a dire, se A è un problema di massimizzazione) distinguiamo i seguenti due casi.

1. $E_B(f_L(x), y) \leq \frac{1}{2\alpha_L \beta_L}$: in questo caso abbiamo che

$$\begin{aligned} R_A(x, g_L(x,y)) - 1 &= \frac{E_A(x, g_L(x,y))}{1 - E_A(x, g_L(x,y))} \\ &\leq \frac{\alpha_L \beta_L E_B(f_L(x), y)}{1 - \alpha_L \beta_L E_B(f_L(x), y)} \\ &\leq 2\alpha_L \beta_L (R_B(f_L(x), y) - 1). \end{aligned}$$

2. $E_B(f_L(x), y) > \frac{1}{2\alpha_L\beta_L}$: in questo caso abbiamo $R_B(f_L(x), y) - 1 \geq \frac{1}{2\alpha_L\beta_L}$ e allora

$$R_A(x, T(x)) - 1 \leq r - 1 \leq 2\alpha_L\beta_L(r-1)(R_B(f_L(x), y) - 1)$$

dove la prima disuguaglianza è dovuta al fatto che T è un algoritmo r-approssimato per A.

Possiamo pertanto definire una tripla (f_E, g_E, α_E) nel modo seguente:

1. Per ogni $x \in I_A$, $f_E(x) = f_L(x)$.
2. Per ogni $x \in I_A$ e per ogni $y \in \mathsf{sol}_B(f_E(x))$,

$$g_E(x,y) = \begin{cases} g_L(x,y) & \text{if } \mathsf{m}_A(x, g_L(x,y)) \geq \mathsf{m}_A(x, T(x)), \\ T(x) & \text{altrimenti.} \end{cases}$$

3. $\alpha_E = \max\{2\alpha_L\beta_L, 2\alpha_L\beta_L(r-1)\}$.

Dalla discussione precedente segue che (f_E, g_E, α_E) è una E-riduzione da A a B. □

In [80] è stato dimostrato che Max 3Sat è APX PB-completo rispetto alla E-riducibilità. Usando il Teorema 3.3 e il fatto che Max 3Sat è in Max SNP [94], è possibile derivare dalla letteratura molti altri risultati di completezza.

Corollario 3.4 *Se un problema A è* Max SNP-*hard rispetto alla L-riducibilità, allora è anche* APX PB-*hard rispetto alla E-riducibilità.*

È chiaro che l'inverso del Teorema 3.3non vale, dal momento che nessun problema in NPO − NPO PB può essere L-ridotto ad un problema in NPO PB, mentre tutti i problemi in PO possono essere E-ridotti ad ogni problema NPO. Ma la L-riducibilità è più restrittiva della E-riducibilità anche all'interno di APX PB.

Teorema 3.5 Max 3Sat *non è* APX PB-*completo rispetto alla L-riducibilità, a meno che* P = NP.

Dimostrazione: Assumiamo che Max 3Sat sia APX PB-completo. allora esiste una L-riduzione (f, g, α, β) da Min Bin Packing a Max 3Sat. Sia x una istanza di Min Bin Packing tale che $\mathsf{opt}(x) \leq 3$, allora $\mathsf{opt}(f(x)) \leq 3\alpha$. È noto che se ϕ è una istanza di Max 3Sat con m clausole, allora $\mathsf{opt}(\phi) \geq m/2$: ne consegue che $f(x)$ ha al più 6α clausole e quindi al più 18α variabili. Una soluzione ottima per $f(x)$ può essere calcolata in tempo $O(6\alpha 2^{18\alpha}) = O(1)$. Da una soluzione ottima per $f(x)$ si ottiene una soluzione ottima per x, e questo implica che Min Bin Packing è risolubile in tempo polinomiale quando viene ristretto ad istanze il cui ottimo valga al più tre. Ma anche questa restrizione di Min Bin Packing è nota essere NP-hard. □

La E-riducibilità è ancora troppo restrittiva, e pare che questo sia vero per qualunque riducibilità nota. Usando la P-riducibilità o la E-riducibilità non sembra possibile dimostrare la APX-completezza di Max 3Sat. La difficoltà è principalmente dovuta al fatto che in queste riducibilità la funzione g mappa soluzioni ottime in soluzioni ottime. La seguente definizione formalizza la proprietà di una riducibilità di preservare l'ottimo.

Definizione 3.6 *Dato un problema* NPO *A e un razionale $r \geq 1$, A_r è la funzione parziale a più valori che, data una istanza x di A, restituisce l'insieme delle soluzioni ammissibili y di x tali che $R(x, y) \leq r$.*

In particolare, $A_1(x) = \mathsf{sol}^*_A(x)$.

Definizione 3.7 *Dato un problema* NPO A *e un razionale* $r \geq$, *un linguaggio* L *appartiene a* P^{A_r} *se esistono due funzioni* f *e* g *calcolabili in tempo polinomiale tali che, per ogni* x, $f(x)$ *è una istanza di* A *con* $\mathsf{sol}(f(x)) \neq \emptyset$, *e, per ogni* $y \in A_r(f(x))$, $g(x,y) = 1$ *se e solo se* $x \in L$. *Introduciamo inoltre la notazione abbreviata* $\mathsf{P}^A \stackrel{\text{def}}{=} \mathsf{P}^{A_1}$.

P^{A_r} può essere visto come l'insieme dei linguaggi che possono essere risolti facendo *una domanda* ad un oracolo che dia soluzioni r-approssimate per istanze di A.

Definizione 3.8 *Sia* $\leq$ *un preordine (cioè una relazione binaria transitiva e riflessiva) in un insieme* $\mathcal{C}$. *Se* $\mathcal{A} \subseteq \mathcal{C}$ *e, per ogni* $b \in \mathcal{A}$, $b \leq a$ *con* $a \in \mathcal{A}$, *allora* a *si dice un* $\leq$-massimo in $\mathcal{A}$.

Definizione 3.9 *Sia* $\leq$ *un preordine in* NPO, *allora* $\leq$ *si dice* optimum-preserving *se* $A \leq B$ *implica* $\mathsf{P}^A \subseteq \mathsf{P}^B$.

È facile vedere che la L-riducibilità, la E-riducibilità, e la P-riducibilità sono dei preordini optimum-preserving in NPO. e che un $\leq$-massimo in APX è un problema APX-completo.

Proposizione 3.10 *Se un esiste un preordine optimum-preserving* $\leq$ *in* NPO *tale che un problema* A *polinomialmente limitato è un* $\leq$*-massimo in* APX, *allora* $\mathsf{P}^{\mathsf{NP}} = \mathsf{P}^{\mathsf{NP}[O(\log n)]}$.

Dimostrazione: Ricordiamo che, da un risultato di [82] segue che $\mathsf{P}^{\mathsf{NP}} \subseteq \mathsf{P}^{\mathrm{MK}}$, dove MK sta per Max Knapsack, inoltre è be noto che Knapsack appartiene a PTAS, e dunwue ad APX [69]. Se A è un $\leq$-massimo in APX, allora Max Knapsack $\leq A$, così che $\mathsf{P}^{\mathsf{NP}} \subseteq \mathsf{P}^{\mathrm{MK}_1} \subseteq \mathsf{P}^A$ (dato che $\leq$ è optimum-preserving). Mostreremo ora che $\mathsf{P}^A \subseteq \mathsf{P}^{\mathsf{NP}[O(\log n)]}$.

Sia L un linguaggio in P^A e siano f e g le funzioni che testimoniano che $L \in \mathsf{P}^A$. Dato che A è polinomialmente limitato e f è calcolabile in tempo polinomiale, allora esiste un polinomio q tale che, per ogni x, $\mathsf{opt}(f(x)) \leq q(|x|)$. Applicando la tecnica di ricerca binaria, è facile vedere che, per ogni x, $\mathsf{opt}(f(x))$ è clacolabile in tempo polinomiale facendo al più $\lceil \log q(|x|) \rceil$ domande ad un oracolo NP. Inoltre, abbiamo che, per ogni x, x appartiene ad L se e solo se esiste una soluzione y per $f(x)$ tale che $\mathsf{m}(f(x),y) = \mathsf{opt}(f(x))$ e $g(x,y) = 1$. In altre parole, dato il valore di $\mathsf{opt}(f(x))$, decidere se $x \in L$ è un problema NP che può essere risolto con una ulteriore domanda all'oracolo, dunque $L \in \mathsf{P}^{\mathsf{NP}[\lceil \log q(n) \rceil + 1]}$.

In conclusione, $\mathsf{P}^{\mathsf{NP}} \subseteq \mathsf{P}^A \subseteq \mathsf{P}^{\mathsf{NP}[O(\log n)]}$.

□

Corollario 3.11 Max Cut, Max Sat, Min Δ-TSP, Min Vertex Cover *non sono* APX-*completi rispetto alla E-riducibilità, né rispetto alla P-riducibilità, a meno che* $\mathsf{P}^{\mathsf{NP}} = \mathsf{P}^{\mathsf{NP}[O(\log n)]}$.

La possibilità che $\mathsf{P}^{\mathsf{NP}} = \mathsf{P}^{\mathsf{NP}[O(\log n)]}$ sembra essere alquanto improbabile e, anche se tale evento non è attualmente correlato ad altri collassi in complessità strutturale, si ritiene che esso implichi il collasso della gerarchia polinomiale [64, 108].

3.1.3 La AP-riducibilità

Gli inconvenienti della E-riducibilità evidenziati in precedenza sono dovuti al fatto di essere optimum-preserving. In effetti, la relazione lineare tra i rapporti di prestazione sembra essere troppo restrittiva. Per superare questo problema non è sufficiente esprimere tale relazione in termini di implicazione, come nella P-riducibilità. Intutivamente, dato che la funzione g *non sa* che approssimazione è richiesta, deve necessariamente mappare soluzioni ottime in soluzioni ottime.

Il nuovo approccio che proponiamo in questa sezione consiste nel consentire che le funzioni f e g *dipendando dal rapporto di prestazione* che la riduzione deve conservare.

Ci sono due vincoli che devono essere imposti sul tempo di calcolo di f e g: vogliamo che la riduzione conservi l'appartenenza a PTAS (perciò il tempo di calcolo deve essere polinomiale per ogni valore fissato del rapporto di prestazione) e vogliamo anche che la riduzione rimanga efficiente anche quando si richiedono rapporti di prestazione molto cattivi (per preservare l'appartenenza a logAPX e polyAPX), perciò il tempo di calcolo non deve aumentare quando il rapporto di prestazione diminuisce. Queste considerazioni conducono alla seguente generalizzazione della nozione di schema di riduzione.

Definizione 3.12 (Schema di riduzione esteso) *Siano A e B due problemi* NPO*; uno* schema di riduzione esteso *(f, g) tra loro è una coppia di funzioni che soddisfini le seguenti proprietà.*

1. *Per ogni $x \in I_A$, e per ogni razionale $r > 1$, $f(x, r) \in I_B$, ed è calcolabile in tempo $t_f(|x|, r)$. Inoltre, se $\mathsf{sol}(x) \neq \emptyset$ allora anche $\mathsf{sol}(f(x, r)) \neq \emptyset$.*

2. *Per ogni $y' \in \mathsf{sol}(f(x, r))$, $g(x, y', r) \in \mathsf{sol}(x)$ è calcolabile in tempo $t_g(|x|, |y|, r)$.*

3. *Per ogni r fissato, sia $t_f(\cdot, r)$ che $t_g(\cdot, \cdot, r)$ sono limitati da un polinomio.*

4. *Per ogni n ed m fissati, sia $t_f(n, \cdot)$ che $t_g(n, m, \cdot)$ sono funzioni non crescenti.*

Secondo la definizione precedente, funzioni come $2^{1/(r-1)} n^h$ o $n^{1/(r-1)}$ sono limitazioni ammissibili per il tempo di calcolo di f e g, mentre ciò non è vero per funzioni come n^r o 2^n.

La definizione di PTAS-riducibilità (rispettivamente, di AP-riducibilità) è molto simile a quella di P-riducibilità (rispettivamente, di E-riducibilità): la differenza principale è l'uso di schemi di riduzione estesi.

Definizione 3.13 (PTAS-riducibilità) *Siano A e B be due problemi* NPO. *Diremo che A è* PTAS-riducibile *a B, in simboli $A \leq_{\mathrm{PTAS}} B$, se esiste uno schema di riduzione esteso (f, g) tra A e B e una funzione calcolabile $c : (1, \infty) \cap \mathcal{Q}^+ \to (1, \infty) \cap \mathcal{Q}^+$ tali che, per ogni $x \in I_A$, per ogni $y \in \mathsf{sol}_B(f(x))$, per ogni razionale $r > 1$,*

$$R_B(f(x, r), y) \leq c(r) \text{ implica } R_A(x, g(x, y, r)) \leq r \ .$$

La tripla (f, g, c) si dice una PTAS-riduzione da A a B.

Definizione 3.14 (AP-riducibilità) *Siano A e B due problemi* NPO. *Diremo che A è* AP-riducibile *a B, in simboli $A \leq_{\mathrm{AP}} B$, se esiste uno schema di riduzione esteso (f, g) tra A e B e una costante positiva $\alpha > 0$ tali che, per ogni $x \in I_A$, per ogni razionale $r > 1$, e per ogni $y \in \mathsf{sol}_B(f(x, r))$,*

$$R_B(f(x, r), y) \leq r \text{ implica } R_A(x, g(x, y, r)) \leq 1 + \alpha(r - 1).$$

La tripla (f, g, α) si dice una AP-riduzione da A a B.

Talvolta chiameremo (f, g, α) una *α-AP-riduzione* da A a B, e scriveremo $A \leq_{\mathrm{AP}}^{\alpha} B$

Si noti che la AP-riducibilità è una generalizzazione della E-riducibilità. Inoltre, è facile vedere che, contrariamente alla E-riducibilità, ogni problema in PTAS è AP-riducibile ad ogni problema NPO.

3.2 Completezza

3.2.1 Problemi NPO-completi

Mostreremo in questa sezione che esistono problemi naturali che sono completi nella classe NPO; nella prossima sezione tratteremo problemi NPO PB-completi. In precedenza, erano stati ottenuti risultati di completezza solamente per Max NPO, Min NPO, Max PB, e Min PB [35, 92, 22, 74]. Un esempio di tali risultati è il seguente teorema.

Teorema 3.15 Min Ones SAT *è* Min NPO*-completo e* Max Ones SAT *è* Max NPO*-completo, anche se solo un sottoinsieme $\{v_1, \ldots, v_s\}$ delle variabili ha peso non nullo $w(v_i) = 2^{s-i}$ e ogni assegnamento che soddisfa l'istanza da valore vero ad almeno un v_i.*

Costruiremo AP-riduzioni da problemi di massimizzazione a problemi di minimizzazione e viceversa. Usando queste riduzioni mostreremo che un problema che sia Max NPO-completo o Min NPO-completo è in effetti completi per tutto NPO.

Teorema 3.16 Min Ones SAT *e* Max Ones SAT *sono* NPO*-completi.*

Dimostrazione: Per stabilire la NPO-completezza di Min Ones SAT dobbiamo solamente mostrare che esiste una AP-riduzione da un problema Max NPO-completo a Min Ones SAT. Come problema Max NPO-completo useremo la versione ristretta di Max Ones SATdel Teorema 3.15.

Sia x una istanza di Max Ones SAT, cioè una formula ϕ su variabili $v_1, \ldots, v_s$ con pesi $w(v_i) = 2^{s-i}$ e altre varaibili con peso zero. Mostreremo prima una semplice riduzione che conserva l'approssimabilità entro un fattore 2, e poi la modificheremo in modo da ottenere una AP-riduzione.

Sia $f(x)$ la formula $\phi \wedge \alpha_1 \wedge \cdots \wedge \alpha_s$ dove α_i è la forma normale congiuntiva di $(z_i \equiv (\overline{v}_1 \wedge \cdots \wedge \overline{v}_{i-1} \wedge v_i))$, dove $z_1, \ldots, z_s$ sono variabili nuove con peso $w(z_i) = 2^i$ e dove tutte le altre variabili (anche le variabili v_i) hanno peso zero. Se y è un assegnamento che soddisfa $f(x)$, sia $g(x, y)$ la restrizione dell'assegnamento alle variabili che occorrono in ϕ. Questo assegnamento chiaramente soddisfa ϕ.

Si noti che esattamente una della variabili z è vera in un qualunque assignamento che soddisfi $f(x)$. Infatti, se tutte le variabili z fossere false, allora tutte le variabili v sarebbero false, e ϕ non sarebbe soddisfatta. D'altra parte, se due diverse variabili z_i e z_j fossero vere, con $j > i$, allora v_i dovrebbe essere simultaneamente vero e falso, il che è una contraddizione. Dunque,

$$
\begin{aligned}
\mathsf{m}(f(x), y) = 2^i \quad &\Leftrightarrow \quad z_i = 1 \\
&\Leftrightarrow \quad v_1 = v_2 = \cdots = v_{i-1} = 0, v_i = 1 \\
&\Leftrightarrow \quad 2^{s-i} \leq \mathsf{m}(x, g(x, y)) < 2 \cdot 2^{s-i} \\
&\Leftrightarrow \quad \frac{2^s}{\mathsf{m}(f(x), y)} \leq \mathsf{m}(x, g(x, y)) < 2 \frac{2^s}{\mathsf{m}(f(x), y)}.
\end{aligned}
$$

In particulare, ciò è vero per la soluzione ottima. Perciò il rapporto di prestazione per MAX ONES SAT è

$$R(x, g(x, y)) = \frac{\mathsf{opt}(x)}{\mathsf{m}(x, g(x, y))} < \frac{2\dfrac{2^s}{\mathsf{opt}(f(x))}}{\dfrac{2^s}{\mathsf{m}(f(x), y)}} = 2\frac{\mathsf{m}(f(x), y)}{\mathsf{opt}(f(x))} = 2R(f(x), y),$$

il che significa che la riduzione conserva l'approssimabilità entro un fattore moltiplicativo 2.

Estendiamo adesso la costruzione in modo da ottenere $R(x, g(x, y)) \leq (1 + 2^{-k})R(f_k(x), y)$ per ogni intero non negativo k. La riduzione descritta in precedenza corrisponde a $k = 0$.

Per ogni $i \in \{1, \ldots, s\}$ e per ogni $(b_1, \ldots, b_{k(i)}) \in \{0, 1\}^{k(i)}$ dove $k(i) = \min\{s - i, k\}$, abbiamo una variabile $z_{i,b_1,\ldots,b_{k(i)}}$. Sia

$$f_k(x) = \phi \wedge \bigwedge_{\substack{i \in \{1,\ldots,s\} \\ (b_1,\ldots,b_{k(i)}) \in \{0,1\}^{k(i)}}} \alpha_{i,b_1,\ldots,b_{k(i)}},$$

dove $\alpha_{i,b_1,\ldots,b_{k(i)}}$ è la forma normale congiuntiva di

$$\left(z_{i,b_1,\ldots,b_{k(i)}} \equiv \left(\overline{v}_1 \wedge \cdots \wedge \overline{v}_{i-1} \wedge v_i \wedge (v_{i+1} = b_1) \wedge \cdots \wedge (v_{i+k(i)} = b_{k(i)})\right)\right).$$

Definiamo $g(x, y)$ come prima. Infine, definiamo

$$w(z_{i,b_1,\ldots,b_{k(i)}}) = \left\lceil \frac{K \cdot 2^s}{w(v_i) + \sum_{j=1}^{k(i)} b_j w(v_{i+j})} \right\rceil = \left\lceil \frac{K \cdot 2^i}{1 + \sum_{j=1}^{k(i)} b_j 2^{-j}} \right\rceil$$

(scegliendo K maggiore di 2^k possiamo trascurare l'effetto dell'operatore di parte intera superiore nei calcoli seguenti).

Come nella riduzione precedente, esattamente una delle variabili z è vera in un qualunque assegnamento che soddisfa $f_k(x)$. Se, in una soluzione y per $f_k(x)$, $z_{i,b_1,\ldots,b_{k(i)}} = 1$, allora abbiamo $\mathsf{m}(f_k(x), y) = w(z_{i,b_1,\ldots,b_{k(i)}})$ e sappiamo che

$$\mathsf{m}(x, g(x, y)) \geq w(v_i) + \sum_{j=1}^{k(i)} b_j w(v_{i+j}) = 2^{s-i}(1 + \sum_{j=1}^{k(i)} b_j 2^{-j}).$$

D'altra parte, se $k(i) = s - i$ allora $\mathsf{m}(x, g(x, y)) = 2^{s-i}(1 + \sum_{j=1}^{k(i)} b_j 2^{-j})$, altrimenti

$$\mathsf{m}(x, g(x, y)) \leq w(v_i) + \sum_{j=1}^{k} b_j w(v_{i+j}) + \sum_{j=k+i+1}^{s} w(v_j) < 2^{s-i}(1 + \sum_{j=1}^{k} b_j 2^{-j})(1 + 2^{-k}).$$

In entrambi i casi, otteniamo

$$\frac{K \cdot 2^s}{\mathsf{m}(f_k(x), y)} \leq \mathsf{m}(x, g(x, y)) < \frac{K \cdot 2^s}{\mathsf{m}(f_k(x), y)}(1 + 2^{-k})$$

e perciò $R(x, g(x, y)) < (1 + 2^{-k})R(f_k(x), y)$. Dato comunque $r > 1$, se scegliamo k tale che $2^{-k} \leq (r - 1)/r$, ad esempio $k = \lceil \log r - \log(r - 1) \rceil$, allora $R(f_k(x), y) \leq r$ implica $R(x, g(x, y)) < (1 + 2^{-k})R(f_k(x), y) \leq r + r2^{-k} \leq r + r - 1 = 1 + 2(r - 1)$. Questa è ovviamente una AP-riduzione con $\alpha = 2$.

Una prova molto simile mostra che MAX ONES SAT è NPO-complete.

$\square$

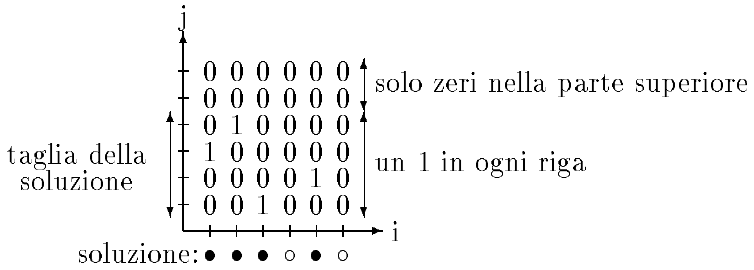


Figura 3.1: L'idea della riduzione da Minimum/Max PB $0-1$ Programming a Maximum/Min PB $0-1$ Programming. La variabile $x_i^j = 1$ se e solo se v_i la j-esima 1-variabile nella soluzione. C'è al più un 1 in ogni colonna e in ogni riga.

Corollario 3.17 *Ogni problema* Min NPO*-completo è* NPO*-completo e ogni problema* Max NPO*-completo è* NPO*-completo.*

3.2.2 Problemi NPO PB-complete

Possiamo anche mostrare che esistono problemi naturali completi nella classe dei problemi NPO limitati polinomialmente.

Teorema 3.18 Max PB $0-1$ Programming *e* Min PB $0-1$ Programming *sono* NPO PB*-completi.*

Dimostrazione: Max PB $0-1$ Programming è noto essere Max PB-completo [22] e Min PB $0-1$ Programming è noto essere Min PB-completo [74]. Dunque, dobbiamo solo mostrare che esistono AP-riduzioni da Min PB $0-1$ Programming a Max PB $0-1$ Programming e da Max PB $0-1$ Programming a Min PB $0-1$ Programming.

Entrambe le riduzioni usano esattamente la stessa costruzione. Dato un assegnamento alla variabili, definiamo le 1-variabili come le variabili che occorrono nella funzione obiettivo che hanno peso 1. Il valore della funzione obiettivo è il numero di 1-variabili più 1.

La misura di una soluzione è codificata introducendo un ordine sulle 1-variabili. L'ordine è codificato da un numero quadratico di variabili $0-1$, si veda la Figura 3.1. L'idea è di invertire le misure, cosi che una soluzione senza 1-variabili corrisponda ad una misura n nel nuovo problema, e, in generale, una soluzione con p 1-variabili corrisponda ad una misura $\left\lfloor \frac{n}{p+1} \right\rfloor$.

Le riduzioni sono costruite nel modo seguente. Data una istanza di Min PB $0-1$ Programming o Max PB $0-1$ Programming, cioè una funzione obiettivo $1 + \sum_{k=1}^{m} v_k$ e alcune disuguaglianze sulle variabili $V = \{v_1, \ldots, v_m\} \cup U$, si costruiscano m^2 variabili x_i^j, $1 \le i, j \le m$, e le seguenti disuguaglianze:

$$\forall i \in [1..m] \qquad \sum_{j=1}^{m} x_i^j \le 1 \qquad \text{(al più un 1 in ogni colonna)} \tag{3.1}$$

$$\forall j \in [1..m] \qquad \sum_{i=1}^{m} x_i^j \le 1 \qquad \text{(al più un 1 in ogni riga)} \tag{3.2}$$

$$\forall j \in [1..m-1] \quad \sum_{i=1}^{m} x_i^j - \sum_{i=1}^{m} x_i^{j+1} \ge 0 \quad \text{(solo zeri nella parte superiore)} \tag{3.3}$$

Oltre a queste disugaglianze, includiamo tutte le disuguaglianze del problema originale, ma sostituiamo ogni variabile v_i con la somma $\sum_{k=1}^{m} x_i^k$. Le variabili in U (che non occorrono nella funzione obiettivo) sono lasciate intatte.

La funzione obiettivo è definita come

$$n - \sum_{p=1}^{m} \left(\left\lfloor \frac{n}{p} \right\rfloor - \left\lfloor \frac{n}{p+1} \right\rfloor \right) \sum_{i=1}^{m} x_i^p. \tag{3.4}$$

Per esprimere la funzione obiettivo solamente con coefficienti binari introduciamo n nuove variabili $y_1, \ldots, y_n$ dove $y_j = 1 - \sum_{i=1}^{m} x_i^p$ per $\lfloor n/(p+1) \rfloor < j \leq \lfloor n/p \rfloor$ e $y_j = 1$ per $j \leq \lfloor n/(m+1) \rfloor$. La funzione obiettivo diventa dunque $\sum_{j=1}^{n} y_j$. È adesso possibile verificare che una soluzione per una istanza del problema originale con s 1-variabili (cioè con misura $s+1$) corrisponde esattamente ad una soluzione nell'istanza del nuovo problema con misura $\lfloor n/(s+1) \rfloor$ e viceversa.

Supponiamo che la soluzione ottima dell'istanza di partenza avesse M 1-variabili, allora il rapporto di prestazione $(s+1)/(M+1)$ corrisponderà al rapporto di prestazione

$$\frac{\left\lfloor \frac{n}{M+1} \right\rfloor}{\left\lfloor \frac{n}{s+1} \right\rfloor} = \frac{s+1}{M+1} \left(1 \pm \frac{m}{n} \right)$$

per il nuovo problema, dove $\frac{m}{n}$ è l'errore relativo dovuto all'operazione di parte intera. Scegliendo n sufficientemente grande l'errore relativo può essere reso arbitrariamente piccolo. È dunque facile vedere che la riduzione è una AP-riduzione.

□

Corollario 3.19 *Ogni problema* Min PB-*completo è* NPO PB-*completo e ogni problema* Max PB-*completo è* NPO PB-*completo.*

3.2.3 Problemi APX- e FAPX-completi

In questa sezione dimostreremo che Max 3Sat è APX-completo. A questo scopo, citiamo il seguente risultato da [80] che segue dalla dimostrazione del teorema PCP.

Teorema 3.20 ([5, 80]) *Esoste una costante $\epsilon > 0$ e due funzioni calcolabili in tempo polinomiale f_s e g_s tali che, per ogni formula booleana ϕ in forma normale congiuntiva, $\psi = f_s(\phi$ è una istanza di* Max 3Sat*' tale che, per ogni assegnamento τ che soddisfa almeno un frazione $1/(1+\epsilon)$ del numero massimo di clausole soddisfacibili in ψ, $g_s(\phi, \tau)$ soddisfa ϕ se e solo se ϕ è soddisfacibile.*

Teorema 3.21 Max 3Sat *è* APX-*complete.*

Dimostrazione: Dal Teorema 2.17 segue che è sufficiente dimostrare che ogni problema di massimizzazione in APX è AP-riducibile a Max 3Sat.

Sia A un problema di massimizzazione in APX e sia T_A un algoritmo r_A-approssimato per A. Il nostro obiettivo è definire una AP-riduzione (f, g, α) da A a Max 3Sat. Prima di tutto, definiamo

$$\alpha = 2(r_A \log r_A + r_A - 1)\frac{1+\epsilon}{\epsilon}$$

dove ϵ è la costante del Teorema 3.20. Per definire le funzioni f e g, per ogni r, distinguiamo due casi.

1. $1+\alpha(r-1) \geq r_A$. In questo caso, per ogni x, definiamo semplicemente

$$g(x, \cdot, r) = T_A(x)$$

il che soddisfa la definizione di AP-riduzione.

2. $r_n = 1+\alpha(r-1) > r_A$. In questo caso abbiamo che

$$r = \frac{r_n - 1}{\alpha} + 1 = \frac{\epsilon}{2(1+\epsilon)} \frac{r_n - 1}{r_A \log r_A + r_A - 1} + 1 \leq \frac{\epsilon}{2k(1+\epsilon)} + 1 \tag{3.5}$$

dove $k = \lceil \log_{r_n} r_A \rceil$ e l'ultima disuguaglianza è dovuta al fatto che

$$k \leq \frac{\log r_A}{\log r_n} + 1 \leq \frac{r_n \log r_A}{r_n - 1} + 1 \leq \frac{r_A \log r_A + r_A - 1}{r_n - 1}.$$

Partizioniamo gli intervalli $[t_A(x), r_A t_A(x)]$ dove $t_A(x) = m_A(x, T_A(x))$ nei seguenti k sottointervalli:

$$[t_A(x), r_n t_A(x)], [r_n t_A(x), r_n^2 t_A(x)], \ldots, [r_n^{k-1} t_A(x), r_A t_A(x)].$$

Si noti che $\mathsf{opt}_A(x) \leq r_A t_A(x) \leq r_n^k t_A(x)$, vale a dire, il valore dell'ottimo per x appartiene ad uno di tali sottointervalli. Per ogni $i = 0, \ldots, k-1$, consideriamo il problema NP di decidere se $\mathsf{opt}_A(x) \geq r_n^i t_A(x)$. Usando la dimostrazione del teorema di Cook [32], possiamo trovare una formula booleana ϕ_i tale che dato un assegnamneto τ_i che soddisfi φ_i possiamo calcolare in tempo polinomiale una soluzione $y \in \mathsf{sol}_A(x)$ tale che $\mathsf{m}_A(x, y) \geq r_n^i t_A(x)$. Definiamo allora l'istanza ψ di Max 3Sat nel modo seguente:

$$\psi = f(x, r) = f_s(\phi_0) \wedge \ldots \wedge f_s(\phi_{k-1})$$

dove f_s è la funzione che appare nel Teorema 3.20. Senza perdere in generalità, possiamo assumere che ogni $f_s(\phi_i)$ contenga lo stesso numero m di clausole.

Sia τ un assegnamento alle variabili di ψ il cui rapporto di prestazione sia al più r. Allora

$$\mathsf{opt}(\psi) - \mathsf{m}(\psi, \tau) \leq \mathsf{opt}(\psi) \frac{r-1}{r} \leq km \frac{r-1}{r}.$$

D'altra parte, se r_i denota il rapporto di prestazione di τ rispetto a $f_s(\phi_i)$, abbiamo che

$$\mathsf{opt}(\psi) - \mathsf{m}(\psi, \tau) \geq \mathsf{opt}(f_s(\phi_i)) - \mathsf{m}(f_s(\phi_i), \tau) = \mathsf{opt}(f_s(\phi_i)) \frac{r_i - 1}{r_i} \geq \frac{m}{2} \frac{r_i - 1}{r_i}$$

where the last inequality is due to the fact that at least half clauses of a Boolean formula are always satisfiable.

Dunque, per $i = 0, \ldots, k-1$,

$$\frac{m}{2} \frac{r_i - 1}{r_i} \leq km \frac{r-1}{r},$$

e allora,

$$1 - 2k\frac{r-1}{r} \leq \frac{1}{r_i}.$$

Usando la disuguaglianza 3.5 e semplici manipolazioni algebriche otteniamo

$$r_i \leq 1 + \epsilon.$$

Per ogni $i = 0, \ldots, k-1$, $\tau_i = g_s(\phi_i, \tau)$ soddisfa ϕ_i se e solo se ϕ_i è soddisfacibile. Sia j il massimo i tale che τ_i soddisfa ϕ_i (si osservi che tale j deve esistere). Allora

$$r_n^j t_A(x) \leq \mathsf{opt}_A(x) \leq r_n^{j+1} t_A(x)$$

e da τ_j possiamo derivare una soluzione y la cui misura sia almeno $r_n^j t_A(x)$, cioè, y è r_n-approssimata. Questo soddisfa la definizione di AP-riduzione.

Dato che A era un arbitrario problema di massimizzazione in APX, ne consegue che MAX 3SAT è APX-completo.

□

Abbiamo dimostrato la APX-completezza di MAX 3SAT da zero per dimostrare il potere della AP-riducibilità: la dimostrazione precedente è leggermente oiù semplice della dimostrazione di APX PB-completezza apparsa in [80]. Khanna et al. [80] partizionano l'intervallo $[t_A(x), r_A t_A(x)]$ in un numero polinomiale (anziché costante) di intervalli, e allota sono necessari un pò di calcoli per mostrare che una approssimazione per l'istanza di MAX 3SAT consente di ottenere un'approssimazione per il problema A.

Alternativamente, possiamo usare i risultati di [80] così come sono e mostrare come rimuovere la condizione di limitazione polinomiale.

Teorema 3.22 *Sia A un problema* NPO *$r(n)$-approssimabile. Allora esiste un problema* NPO *B con le seguenti proprietà.*

1. *$A \leq_{\mathrm{AP}} B$.*
2. *B è $r(n)$-approximable.*
3. *Per ogni $x \in I_B$, per ogni $y \in \mathsf{sol}(x)$, $\mathsf{m}(x,y) \leq |x| r(|x|)$.*

DIMOSTRAZIONE: Assumiamo che A sia un problema di massimizzazione. Sia T_A un algoritmo $r(n)$-approssimato per A; per ogni istanza $x \in I_A$ sia $t_A(x) = \mathsf{m}_A(x, T_A(x))$ la misura della soluzione $r(n)$-approssimata trovata da T_A. Definiamo un problema di ottimizzazione B nel modo seguente: $I_B = I_A$, per ogni $x \in I_B$, definiamo $\mathsf{sol}_B(x) = \mathsf{sol}_A(x)$, per ogni istanza x di taglia n e per ogni soluzione ammissibile y, la misura è

$$\mathsf{m}_B(x,y) = \left\lfloor n \frac{\max\{\mathsf{m}_A(x,y), t_A(x)\}}{t_A(x)} \right\rfloor .$$

Notiamo che per ogni $x \in I_B$, $y \in \mathsf{sol}_B(x)$, abbiamo $\mathsf{m}_B(x,y) \leq n\frac{\max\{\mathsf{m}_A(x,y), t_A(x)\}}{t_A(x)} \leq nr(n)$. In particolare, $\mathsf{opt}_B(x) = \left\lfloor n\frac{\mathsf{opt}_A(x)}{T_A(x)} \right\rfloor$. Oltre ad essere un algoritmo $r(n)$-approssimato per A, T_A è anche un algoritmo $r(n)$-approssimato per B.

$$R_B(x, T_A(x)) = \frac{\left\lfloor n\frac{\mathsf{opt}_A(x)}{t_A(x)}\right\rfloor}{n} \leq \frac{\mathsf{opt}_A(x)}{t_A(x)} \leq r(n)$$

Consideriamo una istanza $x \in I_A$ e una soluzione $y \in \mathsf{sol}_A(x)$ tale chet $\mathsf{m}_A(x,y) \geq t_A(x)$, dimostreremo adesso che $R_A(x,y)$ non può essere molto più grande di $R_B(x,y)$.

$$\begin{aligned}
R_B(x,y) &= \frac{\mathsf{opt}_B(x)}{m_B(x)} \\
&= \frac{\left\lfloor n\frac{\mathsf{opt}_A(x)}{t_A(x)}\right\rfloor}{\left\lfloor n\frac{\mathsf{m}_A(x,y)}{t_A(x)}\right\rfloor} \\
&\geq \frac{n\frac{\mathsf{opt}_A(x,y)}{t_A(x)} - 1}{n\frac{\mathsf{m}_A(x)}{t_A(x)}} \\
&= \frac{\mathsf{opt}_A(x,y)}{\mathsf{m}_A(x)} - \frac{t_A(x)}{n\mathsf{m}_A(x)} \\
&\geq R_A(x,y) - \frac{1}{n}
\end{aligned}$$

Se A è un problema di minimizzazione, definiamo B in modo simile, eccetto che per la misura.

$$\mathsf{m}_B(x,y) = \left\lceil nr(n)\frac{\min\{\mathsf{m}_A(x,y), t_A(x)\}}{t_A(x)}\right\rceil .$$

Abbiamo chem per ogni istanza x di taglia n, $m_B(x,y) \leq |x|r(|x|)$ e $\mathsf{opt}_B(x) = \left\lceil nr(n)\frac{\mathsf{opt}_A(x)}{t_A(x)}\right\rceil$. Come prima, T_A è un algoritmo $r(n)$-approssimato per B, dato che

$$R_B(x, T_A(x)) = \frac{nr(n)}{\left\lceil nr(n)\frac{\mathsf{opt}_A(x)}{t_A(x)}\right\rceil} \leq \frac{t_A(x)}{\mathsf{opt}_A(x)} \leq r(n)$$

Sia ora $x \in I_A$ e sia $y \in \mathsf{sol}_A(x)$ una soluzione tale che $m_A(x,y) \leq t_A(x)$, allora abbiamo che

$$\begin{aligned}
R_B(x,y) &= \frac{m_B(x)}{\mathsf{opt}_B(x)} \\
&= \frac{\left\lceil nr(n)\frac{\mathsf{m}_A(x,y)}{T_A(x)}\right\rceil}{\left\lceil nr(n)\frac{\mathsf{opt}(x)}{T_A(x)}\right\rceil} \\
&\geq \frac{nr(n)\frac{\mathsf{m}_A(x)}{t_A(x)}}{nr(n)\frac{\mathsf{opt}_A(x)}{t_A(x)} - 1} \\
&= \frac{\mathsf{m}_A(x)}{\mathsf{opt}_A(x,y)}\left(\frac{1}{1 - \frac{t_A(x)}{nr(n)\mathsf{opt}_A(x,y)}}\right) \\
&\geq R_A(x,y)\left(\frac{1}{1 - \frac{1}{nr(n)}}\right)
\end{aligned}$$

$$\begin{aligned} &= R_A(x,y)\left(1+\frac{1}{nr(n)-1}\right) \\ &\geq R_A(x,y)+\frac{1}{n-1} \end{aligned}$$

Definiamo adesso la AP-riduzione da A a B.

1. Per ogni $r > 1$, $f(x,r) = x$.
2. Per ogni $x \in I_A = I_B$, per ogni $y \in \mathsf{sol}_B(x)$ e per ogni $r > 1$,

$$g(x,y,r) = \begin{cases} y' & \text{if } r > 1 + 1/(n-1), \\ y^* & \text{altrimenti} \end{cases}$$

   dove $y^* \in \mathsf{sol}^*_B(x)$, mentre $y' = y$ se $R_A(x,y) \leq R_A(x, T_A(x))$, e $y' = T_A(x)$ altrimenti.

3. $\alpha = 2$.

Si noti che, in accordo con la definizione di AP-riducibilità, il tempo di calcolo di g può essere esponenziale in $1/(r-1)$. Se $r \leq 1 + 1/(n-1)$, allora g ha abbastanza tempo per calcolare y^* così che, in questo caso, la definizione di AP-riduzione è soddisfatta.

Consideriamo adesso il caso $r > 1 + 1/(n-1)$, e sia x una istanza e y una soluzione tale che $R_B(x,y) \leq r$. Sia inoltre y' come definita in precedenza. Si ha

$$R_B(x,y') \leq r + \frac{1}{n-1} \leq r + (r-1) = 1 + 2(r-1)\ .$$

Anche in questo caso, la definizione di AP-riduzione è soddisfatta, e questo conclude la dimostrazione. □

Corollario 3.23 *Sia $\mathcal{C} \in \{\mathsf{APX}, \log\mathsf{APX}, \mathrm{poly}\mathsf{APX}\}$, sia A $\mathcal{C}$* PB*-completo rispetto alla E-riducibilità. Allora A è $\mathcal{C}$-completo rispetto alla AP-riducibilità.*

Khanna et al. [80] hanno dimostrato che MAX CLIQUE e MIN NODE COLORING sono polyAPX PB-completi rispetto alla E-riducibilità. FDalla tecnica di riduzione di Khanna et al. [80] e il recente risultato di non-approssimabilità per MIN SET COVER di Raz e Safra [100, 101] (vedi anche un simile risultato indipendente di Arora e Sudan [7]), segue che MIN SET COVER, MIN DOMINATING SET e MIN HITTING SET sono logAPX PB-completi rispetto alla E-riducibilità.

Usando il Corollario 3.23, abbiamo che MAX CLIQUE e MIN NODE COLORING sono polyAPX-completi rispetto alla AP-riducibilità e che MIN SET COVER, MIN DOMINATING SET, e MIN HITTING SET sono logAPX-completi rispetto alla AP-riducibilità.

3.3 Un risultato di completezza senza PCP

In questa sezione dimostreremp che MAX 3SAT è Max NP-completo senza usare il teorema PCP. La nostra riduzione sarà sufficientemente efficiente da avere applicazioni algoritmiche.

3.3.1 La tecnica di base

Illustriamo la nostra tecnica di riduzione presentando una riduzione approximation-preserving da Max Sat a Max kSat. Il valore di k dipende dal fattore di approssimazione che si vuole preservare.

Lemma 3.24 *Sia $p \in (0, 1/2)$, $k \in \mathbf{Z}^+$, $r, r' \geq 1$ tali che*

$$r \geq \max\{1/(1-(1-p)^k)\ ,\ r'/(1-p)\}\ .$$

Allora r-approssimare Max Sat *è riducibile a r'-approssimare* Max kSat.

Dimostrazione: Sia ϕ una istanza di Max Sat con m clausole, sia m_k il numero di clausole di ϕ con al più k letterali, e ϕ_k l'istanza di Max kSat che contiene queste m_k clausole. Definiamo inoltre $m_l = m - m_k$ e $\phi_l = \phi - \phi_k$. Sia τ_k una soluzione r'-approssimata per ϕ_k, sia a il numero della clausole soddisfatte da τ_k. È immediato verificare che

$$\mathsf{opt}(\phi) \leq r'a + m_l\ . \tag{3.6}$$

Infatti, ogni assignamento non può soddisfare più di $r'a$ clausole in ϕ_k (altrimenti τ_k non sarebbe r'-approssimato) a più di m_l clausole in ϕ_l. Definiamo adesso un assegnamento probabilistico τ_R sulle variabili di ϕ. Per ogni variabile x,

$$\mathbf{Pr}[\tau_R(x) \neq \tau_k(x)] = p \quad \mathbf{Pr}[\tau_R(x) = \tau_k(x)] = 1 - p\ .$$

Se una variabile x occurre in ϕ ma non in ϕ_k, allora le assegnamo valore vero o falso con probabilità 1/2.

Stimiamo adesso il numero medio di clausole di ϕ che sono soddisfatte da τ_R. Ogni letterale è vero con probabilità almeno p, perciò la probabilità che una clausola in ϕ_l sia contraddetta è al più $(1-p)^{k+1}$. D'altra parte, se una clausola è soddisfatta da τ_k, allora c'è una probabilità almeno $(1-p)$ che sia ancora soddisfatta da τ_R. Possiamo dunque derivare un limite inferiore al numero di clausole di ϕ che sono soddisfatte in media da τ_R.

$$\mathbf{E}[\mathsf{m}(\phi, \tau_R)] \geq (1-p)a + (1-(1-p)^{k+1})m_l \geq (1/r)\mathsf{opt}(\phi)$$

dove l'ultima disuguaglianza è dovuta alla limitazione (3.6). Usando il metodo delle probabilità condizionali [4] (vedi anche [110]) possiamo trovare in tempo lineare un assegnamento τ tale che $\mathsf{m}(\phi, \tau) \geq \mathsf{m}(\phi, \tau_R)$. Il rapporto di prestazione di τ è almeno r. □

Lemma 3.25 *Per ogni costante $B \in \mathbf{Z}^+$, $r > 1$, r-approssimare* Max GSAT-B *è riducibile ad r'-approssimare* Max Bk-CSP *purché esista una costante $p \in (0, 1/2)$ tale che*

$$r \geq \max\left\{\frac{1}{1-(1-p^B)^{k+1}}\ ,\ \frac{r'}{(1-p)^B}\right\}$$

Dimostrazione: Identica alla dimostrazione del Lemma 3.24, con "letterale" sostituito con "congiunto B-ario". Una istanza di Max GSAT-B in cui tutte le clausole siano disgiunzioni di al più k congiunti è chiaramente una istanza di Max Bk-CSP. Diamo i dettagli del calcolo del rapporto di prestazione di τ_R. Se tutti i letterali sono soddisfatti con probabilità almeno p, allora un congiunto B-ario è contraddetto con probabilità al più $(1-p^B)$. Se un congiunto era soddisfatto da τ, allora è soddisfatto da τ_R con probabilità alemno $(1-p)^B$. Il numero medio di clausole soddisfatte da τ_R è allora

$$\mathbf{E}[\mathsf{m}(\phi, \tau_R)] \geq (1-p)^B a + (1-(1-p^B)^{k+1})m_l \geq (1/r)\mathsf{opt}(\phi)\ .$$

Possiamo usare di nuovo il metodo delle probabilità condizionali per la derandomizzazione. □

3.3.2 Il risultato di Max NP-completezza

Teorema 3.26 *Max 3Sat è Max NP-completo rispetto alla PTAS-riducibilità.*

Dimostrazione: Per ogni B in Max NP, diamo una PTAS-riduzione da Max GSAT-B a Max 3Sat: questa riduzione sarà la composizione della riduzione da Max GSAT-B a Max kB-CSP data nel Lemma 3.25 e della riduzione standard da Max kB-CSP a Max 3Sat [94]. Nella prima riduzione, k non è fissato ma dipende dal fattore di approssimazione che si vuole preservare. Questo nonè in contraddizione con la definizione di PATS-riduzione. Per ogni k, il problema Max k-CSP è in Max SNP, e perciò esiste una PTAS riduzione (f_k, g_k, c_k) da Max k-CSP a Max 3Sat. Una analisi dei risultati di [94] ci assicura che, in realtà, esiste una sola tripla di funzioni (f, g, c) tale che per ogni k $(f(\cdot,\cdot,k), g(\cdot,\cdot,\cdot,k), c(\cdot,k))$ è una PTAS-riduzione da Max k-CSP a Max 3Sat.. Sia ϕ una istanza di Max GSAT-B, $r > 1$. Calcoliamo valori k, r', p tali che

$$r \geq \max\left\{\frac{1}{1-(1-p^B)^{k+1}}\ ,\ \frac{r'}{(1-p)^B}\right\}\ .$$

Prendiamo poi ϕ_k (il sottoinsieme di ϕ che contiene le clausole che sono disgiunzioni di al più k termini) che è una istanza di Max kB-CSP. Da una soluzione r'-approssimata per ϕ_k possiamo ricostruire una soluzione r-approssimata per ϕ. Il primo problema può a sua volta essere ridutto a trovare una soluzione $c(r', kB)$-approssimata per l'istanza $f(\phi_k, r', kB)$ di Max 3Sat. □

Mostreremo adesso che, per dimostrare la APX-completezza di Max 3Sat, è necessaria alemno una versione debole del teorema PCP. Nel seguito diremo che un algoritmo T è un *PTAS non costruttivo* per un problema Max NPO A se valgono le seguenti condizioni.

1. For ogni istanza x di A e ogni razionale $r > 1$, $a = T(x, r)$ è un numero reale con la proprietà che $\mathsf{opt}(x)/r \leq a \leq \mathsf{opt}(x)$.

2. Per ogni $r > 1$ fissato, il tempo di calcolo di $T(x, r)$ è polinomiale in $|x|$.

L'approssimazione non costruttiva per problemi di minimizzazione è definita in modo simile. Si noti che se un problema è in PTAS allora ammette un PTAS non costruttivo, ma il contrario non è necessariamente vero (see [38]).

Lemma 3.27 $\mathsf{PCP}(\log n, 1) = \mathsf{P}$ *se e solo se* Max 3Sat *ammette un PTAS non costruttivo.*

Dimostrazione: La parte "se" è una riformulazione della riduzione standard da verificatori PCP a Max 3Sat (see [5]).

L'altra direzione è più interessante. Per ogni $1/2 \leq s < c \leq 1$ sia Gap 3Sat$_{c,s}$ il seguente problema con promessa: data una formula di 3SAT ϕ con m clausole, rigettare se $\mathsf{opt}(\phi) \leq sm$ e accettare se se $\mathsf{opt}(\phi) \geq cm$. Questo problema è in $\mathsf{PCP}_{c,s}[\log, 3]$ (una dimostrazione è un assegnamento, il verificatore prende una clausola a caso ed accetta e e solo se la clausola è soddisfatta dall'assegnamento in questione) e perciò in $\mathsf{PCP}(\log, 1) = \mathsf{P}$. Denotiamo con $T_{c,s}$ la macchina di Turing polinomiale che, ricevuto in input ϕ, accetta quando $\mathsf{opt}(\phi) \geq cm$, rigetta quando $\mathsf{opt}(\phi) \leq sm$, e il cui comportamente è indefinito negli altri casi. Fissiamo $r > 1$; descriveremo un algoritmo non-costruttivo r-approssimato per Max 3Sat. Sia ϕ una formula di 3SAT con m variabili; il suo ottimo ha un valore compreso tra $m/2$ e m. Dividiamo questo intervallo in $k = \lceil 2/\log r \rceil$ sottointervalli, dove l' i-esimo intervallo (per $i = 0, \ldots, k-1$) è $[m\alpha_i, m\alpha_{i+1})$, con $\alpha_i = .5r^{i/2}$. Per ogni i, eseguiamo $T_{\alpha_{i+1},\alpha_i}(\phi)$. Sia j l'indice più grande tale che $T_{\alpha_{j+1},\alpha_j}(\phi)$ accetta. Dalla definizione di $T_{c,s}$ segue che $\mathsf{opt}(\phi) \geq m\alpha_j$ (dato che $T_{\alpha_{j+1},\alpha_j}(\phi)$ accetta) e che $\mathsf{opt}(\phi) \leq \alpha_{j+2}$ (dato che $T_{\alpha_{j+2},\alpha_{j+1}}(\phi)$ rigetta). Ne consegue che α_j è una soluzione non costruttiva r-approssimata per ϕ. □

Il risultato seguente afferma che i problemi APX-completi presumibilmente non ammettono PTAS non costruttivi.

Lemma 3.28 *Se un problema A APX-completo ammette un PTAS non-construttivo, allora* NP = co-NP.

Dimostrazione: Sia (f, g, c) una PTAS-riduzione da Min Bin Packing ad A; sia $r = c(1.4)$. Sia I una istanza di Min Bin Packing tale che la somma delle dimensioni degli elementi sia 2; è facile vedere che l'impacchettamento ottimo per tale istanza userà 2 o 3 bin. Distinguere tra questi due casi è NP-hard (è una riformulazione del problema Partition[1]). Sia $x = f(I, 1.4)$. Dalla definizione di PTAS-riduzione segue che se y è una soluzione r-approssimata per x, allora $g(I, y, 1.4)$ è una soluzione 1.4-approssimata per I (in realta, una soluzione ottima). Un algoritmo NP può allora calcolare in tempo polinomiale una soluzione v non costruttiva r-approssimata per x, e poi indovinare una soluzione y di misura $\mathsf{m}(x, y) \geq v$, calcolare $P = g(I, y, 1.4)$ e accettare se e solo se $\mathsf{m}(I, P) = 3$. È facile vedere che quello che abbiamo appena descritto è un algoritmo NP per il complemento del problema Partition.

□

I due lemmi precedenti implicano il seguente teorema.

Teorema 3.29 *Se* Max 3Sat *è* APX*-completo rispetto alla PTAS-riducibilità, allora* $\mathsf{PCP}(\log n, 1) \not\subseteq \mathsf{P}$ *a meno che* NP = co-NP.

La conclusione del Teorema 3.29 è più debole del teorema PCP in due aspetti: il verificatore non ha completezza 1 e l'esistenza di problemi intrattabili in $\mathsf{PCP}(\log n, 1)$ assume $\mathsf{NP} \neq co\text{-}\mathsf{NP}$ anziché $\mathsf{P} \neq \mathsf{NP}$. Nessuna di queste variazioni sembra rendere il teorema PCP più semplice da dimostrare.

3.3.3 Risultati algoritmici

In questa sezione considereremo le restrizioni *planari* di Max Sat e Max GSAT-B. Useremo la nostra tecnica di riduzione per svikuppare schemi di approssimazione per entrambi i problemi che richiederanno solo tempo lineare. Gli schemi di approssimazione che descriviamo snon una composizione delle riduzioni della Sezione 3.3.1 e degli schemi di approssimazione lineari per Max kSat e Max k-CSP che sono impliciti nelle tecniche di [13, 68, 79].

Ricordiamo la definizione di *istanza planare* di problemi di soddisfacimento di vincoli.

Definizione 3.30 (Grafo di incidenza) *Sia $\mathcal{F}$ una famiglia (possibilmente infinita) di vincoli. Sia ϕ una istanza di* Max Weight $\mathcal{F}$ *su un insieme di variabili X. Il* grafo di incidenza *di ϕ, denotato con $I_\phi = (V, E)$, è definito nel modo seguente:*

[1] Ricordiamo che il problema NP-completo Partition è definito nel modo seguente: dato un insieme U di elementi e una funzione $s : U \to Q \cap (0, 1]$, esiste un sottoinsieme $U' \subseteq U$ tale che

$$\sum_{u \in U'} s(u) = \sum_{u \notin U'} s(u)?$$

- *V ha un v-vertice per ogni variabile $x \in X$ e un f-vertice per ogni vincolo C di ϕ.*
- *Per ogni vincolo C di ϕ e ogni variabile x che occurre in C, c'è un arco tra il vertice per C e il vertice per x.*

Per ogni famiglia di vincoli $\mathcal{F}$, Max Planar $\mathcal{F}$ è la restrizione di Max $\mathcal{F}$ a istanze il cui grafo di incidenza sia planare. Saremo particolarmente interessati a Max Planar k-CSP, Max Planar Sat, e Max Planar GSAT-B.

Khanna e Motwani [79] hanno dimostrato che gli ultimi due problemi hanno un PTAS che trova soluzioni $(1+\epsilon)$-approssimate in tempo $O(n^{1/\epsilon})$. Miglioreremo il tempo di calcolo portandolo a $O(n)$ (la costante nascosta nella notazione O dipenderà da ϵ e, nel caso di Max Planar GSAT-B anche da B). Useremo il fatto che le tecniche di Khanna e Motwani implicano schemi di approssimazione lineari quando sono ristretta a Max kSat e Max k-CSP.

Teorema 3.31 *[[78]] For fixed $\epsilon > 0$,* Max Planar k-CSP *admits an $(1+\epsilon)$-approximate linear time algorithm,*

Teorema 3.32 *Per ogni $\epsilon > 0$ e $B \in \mathcal{Z}^+$,* Max Planar GSAT-B *ammette un algoritmo lineare $(1+\epsilon)$-approssimato.*

Dimostrazione: Siano k, p, ϵ' costainti (dipendenti solo da ϵ e B) tali che

$$1 + \epsilon \geq \max\left\{\frac{1}{1-(1-p^B)^{k+1}}\ ,\ \frac{1+\epsilon'}{(1-p)^B}\right\}\ .$$

Sia ϕ una istanza planare di Max GSAT-B. La costruzione del Lemma 3.25 reduce il problema di $(1+\epsilon)$-approssimare ϕ al problema di $(1+\epsilon')$ approssimare una istanza ϕ_k di Max kB-CSP ottenuta da ϕ rimuovendo tutte le clausole con più di k congiunti. La riduzione può essere implementata in tempo lineare, non incremente la dimensione dell'istanza e conserva la planarità del grafo di incidenza. Un soluzione $(1+\epsilon')$-approssimata per ϕ' può essere trovata in tempo lineare usando il Teorema 3.31. $\square$

3.4 Query Complexity

La query complexity dell'approssimazione non costruttiva di numerosi problemi di ottimizzazione NP-hard è stata studiata e classificata usando le classi di funzioni $\mathsf{PF}^{\mathsf{NP}[\cdot]}$ [28, 29]. Tuttavia, questo approccio non può essere applicato per analizzare la complessità di "costruire" soluzioni approssimate. Per superare questa limitazione, usiamo un nuovo approccio che, essenzialmente, consiste nel considerare quanto "aiuta" un problema di approssimazione a risolvere problemi decisionali. Per ogni problema NPO A conidereremo quella che chiamiamo la *approximate query hierarchy* di A, cioè l'insieme dei linguaggi che possono essere decisi usando una domanda ad un algoritmo r-approssimato per A, per una certa costante $r > 1$.

Definizione 3.33 *La* classe $\mathsf{AQH}(A)$ *è uguale all'unione* $\bigcup_{r>1} \mathsf{P}^{A_r}$.

Il seguente risultato mostra che un algoritmo di approssimazione per un problema APX non aiuta più di un numero costante di domande ad un problema NP. È bene ricordare che, in generale, una soluzione approssimata, anche se non aiuta molto, richiede un numero super-logaritmico di domande per essere calcolata [27].

Proposizione 3.34 *Per ogni problema A in* APX, $\mathsf{AQH}(A) \subseteq \mathsf{QH}$.

DIMOSTRAZIONE: Sia A un problema di massimizzaizone (la dimostrazione per problemi di minimizzazione è simile). Sia T un algoritmo r-approssimato per A, per qualche $r > 1$, e sia $L \in \mathsf{P}^{A_\rho}$ per qualche $\rho > 1$. Esistono allora due funzioni f e g, calcolabili in tempo polinomiale, che testimoniano quest'ultimo fatto. Per ogni x, sia $m = \mathsf{m}(f(x), T(f(x)))$, così che $m \leq \mathsf{opt}(f(x)) \leq rm$. Possiamo allora partizionare l'intervallo $[m, rm]$ in $\lfloor \log_\rho r \rfloor + 1$ sottointervalli

$$[m, \rho m), [\rho m, \rho^2 m), \ldots, [\rho^{\lfloor \log_\rho r \rfloor - 1} m, \rho^{\lfloor \log_\rho r \rfloor} m], [\rho^{\lfloor \log_\rho r \rfloor} m, rm],$$

e cercare il sottointervallo che contiene il valore ottimo (una tecnica simile era stata usta in [28, 29]). Possiamo chiaramente fare questo, tramite ricerca binaria, con al più $\lfloor \log_\rho r \rfloor + 1$ domande ad un oracolo NP. Una ulteriore domanda è sufficiente per sapere se esiste una soluzione ammissibile y la cui misura sia in tale intervallo etale che $g(x, y) = 1$. Dato che y è ρ-approssimata, segue che l può essere decido usando $\lfloor \log_\rho r \rfloor + 2$ domande, e quindi, $L \in \mathsf{QH}$. □

La seguente proposizione, invece, afferma che ogni linguaggio L nella query hierarchy può essere deciso usando solo una domanda ad A_r, dove A è APX-completo e r dipende solo dal livello della query hierarchy a cui L appartiene. Per dimostrare tale proposizione, abbiamo prima bisogno del seguente risultato tecnico.

Lemma 3.35 *Per ogni problema* APX-*completo A e per ogni k, esistono due funzioni f e g calcolabili in tempo polinomiale ed una costante r tali che per ogni k-tupla $(x_1, \ldots, x_k)$ di istanze di* PARTITION, *$x = f(x_1, \ldots, x_k)$ è una istanza di A e se y è una soluzione per x il cui rapporto di prestazione è più piccolo di r, allora $g(x, y) = (b_1, \ldots, b_k)$ dove $b_i \in \{0, 1\}$ e $b_i = 1$ se e solo se $x_i \in$* PARTITION.

DIMOSTRAZIONE: Sia $x_i = (U_i, s_i)$ una istanza di PARTITION per $i = 1, \ldots, k$. Senza perdere in generality, possiamo assumere che gli insiemi U_i siano a due a due disgiunti e che, per ogni i, $\sum_{u \in U_i} s_i(u) = 2$. Sia $w = (U, s, \preceq)$ una istanza di MIN ORDERED BIN PACKING definita nel modo seguente (una costruzione simile è stata usata in [99]).

1. $U = \bigcup_{i=1}^{k} U_i \cup \{v_1, \ldots, v_{k-1}\}$ dove i v_i sono nuovi oggetti.

2. Per ogni $u \in U_i$, $s(u) = s_i(u)$ e $s(v_i) = 1$ per $i = 1, \ldots, k - 1$.

3. Per ogni $i < j \leq k$, per ogni $u \in U_i$, e per ogni $u' \in U_j$, $u \preceq v_i \preceq u'$.

Ogni soluzione ammissibile per w deve essere formato da una sequenza di impacchettamenti di $U_1, \ldots, U_k$ tale che, per ogni i, i bin usati per U_i siano separati dai bin usati per U_{i+1} da un bin completamente riempito da v_i. In particolare, gli impacchettamenti degli U_i in una soluzione ottima useranno solo due o tre bin: due bin saranno usati se e solo se $x_i \in$ PARTITION. La misura ottima è al più $4k - 1$, e perciò una soluzione $(1 + 1/(4k))$-approssimata è una soluzione ottima.

Dato che MIN ORDERED BIN PACKING appartiene ad APX [109] e A è APX-completo, allora esiste una AP-riduzione (f_1, g_1, α) da MIN ORDERED BIN PACKING ad A. Possiamo allora definire $x = f(x_1, \ldots, x_k) = f_1(w, 1 + 1/(4\alpha k))$ e $r = 1 + 1/(4\alpha k)$. Per ogni soluzione y r-approximata per x, al quarta proprietà della AP-riducibilità implica che $z = g_1(x, y, 1 + 1/(4\alpha k))$ è una soluzione $(1 + 1/(4k))$-approssimata per w e dunque anche una soluzione ottima per w. Da z, possiamo facilmente derivare le risposte corrette alle k domande $x_1, \ldots, x_k$. □

Possiamo adesso dimostrare il seguente risultato.

Proposizione 3.36 *Per ogni problema A APX-completo, $\mathsf{QH} \subseteq \mathsf{AQH}(A)$.*

DIMOSTRAZIONE: Sia $L \in \mathsf{QH}$, allora $L \in \mathsf{P}^{\mathsf{NP}[h]}$ per qualche h. È noto (vedi, ad esempio, [16]) che L può essere ridotto al problema di rispondere a $k = 2^{h-1}$ domande non-adattive a NP. Più formalmente, esistono due funzioni t_1 e t_2 calcolabili in tempo polinomiale tali che, per ogni x, $t_1(x) = (x_1, \ldots, x_k)$, dove $x_1, \ldots, x_k$ sono k istanze del problema PARTITION, e per ogni $(b_1, \ldots, b_k) \in \{0,1\}^k$, $t_2(x, b_1, \ldots, b_k) \in \{0,1\}$. Inoltre, se, per ogni j, $b_j = 1$ se e solo se $x_j \in$ PARTITION, allora $t_2(x, b_1, \ldots, b_k) = 1$ se e solo se $x \in L$.

Siano adesso f, g e r le due funzioni e la costante del Lemma 3.35 applicato al problema A e alla costante k. Per ogni x, $x' = f(t_1(x))$ è una istanza di A tale che se y è una soluzione r-approssimata per x', allora $t_2(g(x', y)) = 1$ se e solo se $x \in L$. Dunque, $L \in \mathsf{P}^{A_r}$. □

Combinando le Proposizioni 3.34 e 3.36, abbiamo dunque il seguente teorema.

Teorema 3.37 *Per ogni problema A APX-completo, $\mathsf{AQH}(A) = \mathsf{QH}$.*

Un approccio alternativo per collegare query complexity e approssimabilità consiste nel definire classi di domande limitate composte da *funzioni a più valore* e classificare i problemi di approssimazione direttamente in tali classi. Questo approccio è stato introdotto da Chang [27]. Daremo nel seguito una caratterizzazione dei problemi completi in APX e in polyAPX, rispettivamente, in termini di complessità del problema approssimato (rispetto alle classi di Chang) e di proprietà combinatorie opportunemente definite.

Definizione 3.38 ([27]) $\mathsf{NPF}^{\mathsf{NP}[q(n)]}$ *è la classe delle funzioni parziali a più valori che sono calcolabili con macchine di Turing polinomiali non deterministiche che pongono al più $q(n)$ domande ad un oracolo NP nell'* intero *albero delle computazioni*[2].

L'hardness in queste classi sarà definita usando la seguente riducibilità che è una estensione sia della riducibilità metrica [72] che della riducibilità one-query [36].

Definizione 3.39 (MV-riducibilità [27]) *Siano F e G due funzioni parziali a più valori. Diciamo che F è MV-riducibile a G (in simboli, $F \leq_{\mathrm{mv}} G$) se esistono due algoritmi polinomiali t_1 e t_2 tali che, per ogni x nel dominio di F, $t_1(x)$ è nel dominio di G e, per ogni $y \in G(t_1(x))$, $t_2(x, y) \in F(x)$.*

La proprietà combinatoria usata per caratterizzare i problemi polyAPX-completi è la ben nota self-improvability (vedi, as esempio, [93]).

Definizione 3.40 (Self-improvability) *Un problema A è* self-improvable *se esistono due algoritmi t_1 e t_2 tali che, per ogni istanza x di A e per ogni due razionali $r_1, r_2 > 1$, $x' = t_1(x, r_1, r_2)$ è una istanza di A e, per ogni $y' \in A_{r_2}(x')$, $y = t_2(x, y', r_1, r_2) \in A_{r_1}(x)$. Inoltre, per ogni r_1 e r_2 fissati, il tempo di calcolo di t_1 e t_2 è polinomiale.*

Siamo adesso pronti per enunciare e dimostrare il nostro prima teorema di caratterizzazione.

Teorema 3.41 *Un problema A in* polyAPX *è* polyAPX*-completo se e solo se è self-improvable e A_{r_0} è $\mathsf{NPF}^{\mathsf{NP}[\log\log n + O(1)]}$-hard per qualche $r_0 > 1$.*

[2] Diciamo che una funzione F parziale a più valori è calcolabile da una macchina di Turing non deterministica N se, per ogni x nel dominio di F, esiste una computazione di $N(x)$ che termina e tutte le computazioni di $N(x)$ che terminano restituiscono in output un valore (non necessariamente lo stesso) di $F(x)$.

DIMOSTRAZIONE: Sia A un problema polyAPX-completo. Datp che MAX CLIQUE è self-improvable [55] e polyAPX-completo [80] e dato che l'equivalenza rispetto alla AP-riducibilità conserva la proprietà di self-improvability (vedi [93]), abbiamo che A è self-improvable. È allora sufficiente dimostrare che A_2 è hard per $\mathsf{NPF}^{\mathsf{NP}[\log\log n+O(1)]}$.

Dalla polyAPX-completezza di A abbiamo che MAX CLIQUE $\leq_{\mathrm{AP}}$ A: sia α la costante di questa riduzione. Dal Teorema 12 in [27] abbiamo che ogni funzione F in $\mathsf{NPF}^{\mathsf{NP}[\log\log n+O(1)]}$ è MV-riducibile a MAX CLIQUE$_{1+\alpha}$. Dalla definizione di AP-riducibilità, abbiamo anche che MAX CLIQUE$_{1+\alpha}\leq_{\mathrm{mv}}A_2$ così che F è MV-riducibile ad A_2.

Sia ora A un problema in polyAPX self-improvable tale che, per un certo r_0, A_{r_0} è $\mathsf{NPF}^{\mathsf{NP}[\log\log n+O(1)]}$-hard. Mostreremo che, per ogni problema B in polyAPX, B è AP-riducibile ad A. A questo scopo, introduciamo la seguente funzione parziale a più valori che chiamiamo `multisat`: data in input una sequenza $(\phi_1,\ldots,\phi_m)$ di istanze di SAT con $m \leq \log|(\phi_1,\ldots,\phi_m)|$ e tale che, per ogni i, se ϕ_{i+1} è soddisfacibile allora ϕ_i è soddisfacibile, un possibile output è un assegnamento che soddisfi ϕ_{i^*} dove $i^* = \max\{i : \phi_i \text{è soddisfacibile }\}$. Dalla dimostrazione del Teorema 12 in [27] segue che questa funzione è $\mathsf{NPF}^{\mathsf{NP}[\log\log n+O(1)]}$-completa.

Facendo uso di tecniche simili a quelle della dimostrazione della Proposizione 3.34, è facile vedere che, per ogni $r > 1$ fissato, $t_1^B(\cdot,r)$ e $t_2^B(\cdot,\cdot,r)$ formano una MV-riduzione da B_r a `multisat`. Inoltre, dato che A_{r_0} è $\mathsf{NPF}^{\mathsf{NP}[\log\log n+O(1)]}$-hard, allora esiste una MV-riduzione (t_1^M,t_2^M) da `multisat` a A_{r_0}. Infine, siano t_1^A e t_2^A le funzioni che testimoniano la self-improvability di A. La AP-riduzione da B ad A può essere derivata nel modo seguente:

$$\begin{array}{ccccccc}
x,r & \xrightarrow{t_1^B(x,r)} & x' & \xrightarrow{t_1^M(x')} & x'' & \xrightarrow{t_1^A(x'',r_0,r)} & x''' \\
 & & & & & & \downarrow \\
y & \xleftarrow{t_2^B(x,y',r)} & y' & \xleftarrow{t_2^M(x',y'')} & y'' & \xleftarrow{t_2^A(x'',y''',r_0,r)} & y'''
\end{array}$$

È facile vedere che se y''' è una soluzione r-approssimata perl'istanza x''' di A, allora y è una soluzione r-approssimata per l'istanza x di B. Dunque, B è AP-reducibile ad A con $\alpha = 1$. □

Per caratterizzare i problemi APX-completi, introduciamo la seguente proprietà combinatoria

Definizione 3.42 (Additività lineare) *Un problema A in NPO di dice* linearmente additivo *se esistono una costante β e due algoritmi t_1 e t_2 tali che, per ogni razionale $r > 1$ e per ogni sequenza $x_1,\ldots,x_k$ di istanze di A, $x' = t_1(x_1,\ldots,x_k,r)$ è una istanza di A e, per ogni $y' \in A_{1+(r-1)\beta/k}(x')$, $t_2(x_1,\ldots,x_k,y',r) = y_1,\ldots,y_k$ dove ogni y_i è una soluzione r-approssimata per x_i. Inoltre, il tempo di calcolo di t_1 e t_2 è polinomiale per ogni r fissato.*

Teorema 3.43 *Un problema A in APX è APX-completo se e solo se è linearmente additivo ed esiste una costante r_0 tale che A_{r_0} è $\mathsf{NPF}^{\mathsf{NP}[1]}$-hard.*

DIMOSTRAZIONE: Sia A un problema APX-completo r_A-approssimabile. Dalla dimostrazione della Proposizione 3.36 abbiamo che esiste una costante r_0 tale che A_{r_0} è hard per $\mathsf{NPF}^{\mathsf{NP}[1]}$. Per dimostrare l'additivit='a lineare, fissiamo un qualunque $r > 1$ e siano $x_1,\ldots,x_k$ istanze di A. Senza perdere in generalità, possiamo assumere che $r < r_A$ (altrimenti le k istanze possono essere r-approssimate l'algoritmo r_A-approssimato). Per ogni $i = 1,\ldots,k$ il problema di troavre una soluzione r-approssimata y_i per x_i è riducibile al problema di risolvere costruttivamente un insieme di $\lceil\log_r r_A\rceil$ istanze di PARTITION. Si noti che $\lceil\log_r r_A\rceil \leq c/(r-1)$ per una certa costante c che dipende solo da r_A. Inoltre, possiamo dimostrare che esiste una costante γtale che risolvere costruttivamente $kc/(r-1)$ istanze di PARTITION è riducibile ad $(1+\gamma(r-1)/kc)$-approssimare una singola istanza di A (questo può essere dimostrato lungo le linee della Proposition 3.36). Perciò, A è linearmente additivo con $\beta = \gamma/c$.

Sia ora A un problema in APX linearmente additivo tale che A_{r_0} è $\mathsf{NPF}^{\mathsf{NP}[1]}$-hard per qualche r_0 e sia B un problema r_B-approssimabile. Data una istanza x di B, per ogni $r > 1$ possiamo ridurre il problema di trovare una soluzione r-approssimata per x al problema di risolvere costruttivamente $c/(r-1)$ istance di PARTITION, per una opportuna costante c che non dipende da r. Ognuna di questi problemi decisionali è riducibile ad A_{r_0}, dato che ogni problema NP può essere costruttivamente risolto da una funzione $\mathsf{NPF}^{\mathsf{NP}[1]}$. Dalla additività lineare, se gue che r_0-approssimare $c/(r-1)$ istanze di A è riducibile a $(1+\beta(r_0-1)(r-1)/c)$-approssimare una singola istanza di A. Questa è una AP-riduzione da A a B con $\alpha = c/(\beta(r_0-1))$. □

3.5 Conseguenze per problemi naturali

3.5.1 Problemi naturali APX-intermedi

Il seguente risultato mostra che la Proposizione 3.34 può essere rafforzata quando ci si restringa a considerare problemi in APTAS.

Proposizione 3.44 *Se $A \in \mathsf{APTAS}$, allora esiste una costante h tale che $\mathsf{AQH}(A) \subseteq \mathsf{P}^{\mathsf{NP}[h]}$.*

DIMOSTRAZIONE: Sia A un problema di minimizzazione in APTAS (la dimostrazione per problemi di massimizzazione è molto simile). Per definizione, esistono una costante k e un algoritmo T tali che, per ogni istanza x e per ogni razionale $r > 1$,

$$\mathsf{m}(x, T(x,r)) \leq r \cdot \mathsf{opt}(x) + k.$$

Dimostreremo adesso che esiste una costante h tale che, per ogni $r > 1$, esite una funzione $l_r \in \mathsf{PF}^{\mathsf{NP}[h-1]}$ tale che, per ogni istanza x del problema A,

$$\mathsf{opt}(x) \leq l_r(x) \leq r \cdot \mathsf{opt}(x).$$

Intuitivamente, le funzioni l_r formano uno schema di approssimazione non costruttivo che è calcolabile con un numero costante di domande ad un oraclo NP-completo. Data una istanza x, possiamo verificare se $\mathsf{sol}(x) = \emptyset$ tramite una singola domanda ad un oracolo NP, così che possiamo restringerci a considerare istanze tali che $\mathsf{sol}(x) \neq \emptyset$ (e dunque $\mathsf{opt}(x) \geq 1$). Notiamo che, per tali istanze, $T(\cdot, 2)$ è un algoritmo $(k+2)$-approssimato per A. Fissiamo un $r > 1$, e sia $\epsilon = r-1$, $y = T(x, 1+\epsilon/2)$ e $a = m(x, T(x,2))$. Dobbiamo considerare due casi

1. $a \geq 2k(k+2)/\epsilon$: in questo caso, $\mathsf{opt}(x) \geq 2k/\epsilon$, cioè, $\mathsf{opt}(x)\epsilon/2 \geq k$. Quindi

$$\begin{aligned} \mathsf{m}(x,y) &\leq \mathsf{opt}(x)(1+\epsilon/2) + k \\ &\leq \mathsf{opt}(x)(1+\epsilon/2) + \mathsf{opt}(x)\epsilon/2 \\ &= \mathsf{opt}(x)(1+\epsilon) = r\mathsf{opt}(x), \end{aligned}$$

e dunque y è una soluzion r-approssimata per x, e possiamo definire $l_r(x) = \mathsf{m}(x,y)$ (in questo caso, è stat sufficiente una sola domanda per calcolare l_r).

2. $a < 2k(k+2)/\epsilon$: in questo caso, $\mathsf{opt}(x) < 2k(k+2)/\epsilon$. Dunque,

$$\mathsf{opt}(x) \leq \mathsf{m}(x,y) \leq \mathsf{opt}(x) + \mathsf{opt}(x)\epsilon/2 + k < \mathsf{opt}(x) + k(k+2) + k.$$

Chiaramente, $\lceil \log k(k+3) \rceil$ domande ad NP sono sufficienti per trovare il valore $\mathsf{opt}(x)$ tramite la tecnica di ricerca lorale: in questo caso $l_r(x) = \mathsf{opt}(x)$ è stato calcolato tramite $\lceil \log k(k+3) \rceil + 1$ domande.

Sia ora L un linguaggio in $\mathsf{AQH}(A)$, allora $L \in \mathsf{P}^{A_r}$ per qualche $r > 1$. Siano f e g le funzioni che testimoniano che $L \in \mathsf{P}^{A_r}$. Osserviamo che, per ogni x, $x \in L$ se e solo se esiste una soluzione y per $f(x)$ tale che $\mathsf{m}(f(x), y) \geq l_r(f(x))$ e $g(f(x), y) = 1$: vale a dire, dato $l_r(f(x))$, decidere se $x \in L$ è un problema in NP. Dal momento che $l_r(f(x))$ è calcolabile con al più $\lceil \log k(k+3) \rceil + 1$ domande ad NP, abbiamo che $L \in \mathsf{P}^{\mathsf{NP}[h]}$ dove $h = \lceil \log k(k+3) \rceil + 2$. □

Come conseguenza, esistono problemi naturali APX-intermedi.

Teorema 3.45 *Se la gerarchia polinomiale non collasse, allora* Min Bin Packing, Min Degree Spanning Tree, *e* Min Edge Coloring *sono* APX*-intermedi.*

Dimostrazione: Dalla Proposizione 3.44 e dal fatto che Min Bin Packing è in APTAS[76], segue che $\mathsf{AQH}(\text{Min Bin Packing}) \subseteq \mathsf{P}^{\mathsf{NP}[h]}$ per un certo h. Se Min Bin Packing fosse APX-completo, allora dalla Proposizione 3.36 seguirebbe che $\mathsf{QH} \subseteq \mathsf{P}^{\mathsf{NP}[h]}$. Dal Teorema 2.25 avremmo dunque il collasso della gerarchia polinomiale. Le dimostrazioni per Min Degree Spanning Tree e Min Edge Coloring sono identiche ed usano i risultati di [67, 52] □

Si noti che il risultato precedente sembra non essere ottenibile sotto l'ipotesi che $\mathsf{P} \neq \mathsf{NP}$, come mostrato nel seguente teorema.

Teorema 3.46 *Se* $\mathsf{NP} = \text{co-}\mathsf{NP}$ *allora* Min Bin Packing *è* APX*-completo.*

Dimostrazione: Assumiamo $\mathsf{NP} = co\text{-}\mathsf{NP}$, presenteremo una AP-riduzione da Max Sat a Min Bin Packing. Dato che $\mathsf{NP} = conp$, essite una macchina di Turing polinomiale non deterministica M tale che, presa in input una istanza ϕ di Max Sat, ha almeno una computazione accettante e tutte le computazioni accettanti si fermano con una soluzione ottima per ϕ scritta sul nastro. Infatti, M indovina un intero k, un assegnamento τ tale che $\mathsf{m}(\phi, \tau) = k$ e una dimostrazione del fatto che $\mathsf{opt}(\phi) \leq k$. Dalla dimostrazione del teorema di Cook segue che, data ϕ, è possibile trovare in tempo polinomiale una formula ϕ' che sia soddisfacibile e tale che dato un qualunque assegnamento che soddisfa ϕ' possiamo trovare in tempo polinomiale una soluzione ottima per ϕ. Combinando questa costruzione con la dimostrazione di NP-completezza di Min Bin Packing, otteniamo due funzioni t_1 e t_2 calcolabili in tempo polinomiale tali che, per ogni istanza ϕ di Max Sat, $t_1(\phi) = x_\phi$ è una istanza di Min Bin Packing tale che $\mathsf{opt}(x_\phi) = 2$ e, per ogni soluzione ottima y per x_ϕ, $t_2(x_\phi, y)$ è una soluzione ottima per ϕ. Si noti che, per costruzione, una soluzione r-approssimata per x_ϕ è anche una soluzione ottima, quando $r < 3/2$. Sia T un algoritmo 4/3-approssimato per Max Sat [110, 58]. La riduzione da Max Sat a Min Bin Packing è definita nel modo seguente: $f(\phi, r) = t_1(\phi)$;

$$g(\phi, y, r) = \begin{cases} T(\phi) & \text{se } r \geq 4/3, \\ t_2(t_1(\phi), y) & \text{altrimenti.} \end{cases}$$

È immediato verificare che quella che abbiamo appena definito è una AP-riduzione con $\alpha = 1$. □

3.5.2 Min Set Cover non è self-improvable

È ancora una interessante questione aperta il torvare una proprietà combinatoria che caratterizzi i problemi logAPX-completi, Tale proprietà non può comunque essere la self-improvability.

Teorema 3.47 *Nessun problema* logAPX*-completo può essere self-improvable a meno che la gerarchia polinomiale collassi.*

Dimostrazione: Consideriamo il problema di ottimizzazione Max Number of Satisfiable Formulas (in breve, MNSF) definito nel modo seguente.

Istanza: Un insieme di m formule booleane $\phi_1, \ldots, \phi_m$ in 3CNF, tale che ϕ_1 è una tautologia e $m \leq \log n$, dove n è la dimensione dell'istanza.

Soluzione: Assegnamento τ per le variabili di $\phi_1, \ldots, \phi_m$.

Misura: Il numero di formule soddisfatte, cioè $|\{i : \phi_i \text{ è soddisfatta da } \tau\}|$.

Chiaramente, MNSF è in logAPX, dal momento che la misura di ogni assegnamento è almeno 1, e il valore dell'ottimo è sempre minore o uguale di $\log n$, dove n è la dimensione dell'input. Mostreremo che, per ogni $r < 2$, MNSF$_r$ è hard per $\mathsf{NPF}^{\mathsf{NP}[\log\log\log n - 1]}$.

Date $\log\log n$ domande (di dimensione polinomiale in n) ad un linguaggio NP-completo, possiamo costruire una istanza $\Phi = \phi_1, \ldots, \phi_m$ di MNSF dove ϕ_1 è una tautologia e, per ogni $i \geq 1$, le formule $\phi_{2^i} = \ldots = \phi_{2^{i+1}-1}$ sono soddisfacibili se e solo se almeno i istanze tra $x_1, \ldots, x_{\log\log n}$ sono istanze-si (queste formule sono facili da costruire usando la dimostrazione del teorema di Cook). Notiamo che $m = 2^{\log\log n + 1} - 1$ e, aggiungendo clausole ad alcune formule, possiamo raggiungere la limitazione $m \leq \log|\phi_1, \ldots, \phi_m|$. Inoltre, da una soluzione r-approssimata per Φ, possiamo decidere quante istanze in $x_1, \ldots, x_{\log\log n}$ sono istanze-si, e possiamo inoltre ottenere soluzioni per tali istanze. Dunque, ogni funzione in $\mathsf{NPF}^{\mathsf{NP}[\log\log\log n - 1]}$ è MV-riducibile a MNSF$_r$.

Sia A un problema self-improvable logAPX-completo. Per ogni funzione $F \in \mathsf{NPF}^{\mathsf{NP}[\log\log\log n - 1]}$, $F \leq_{\mathrm{mv}} \mathrm{MNSF}_{1.5} \leq_{\mathrm{mv}} A_{1+1/2\alpha} \leq_{\mathrm{mv}} A_{2^{16}}$ dove α è la costante della AP-riduzione da MNSF ad A e dove l'ultimo riduzione è dovuta alla self-improvability di A. Perciò, per ogni x, calcolare $F(x)$ è riducibile a trovare una soluzione 2^{16}-approssimata per una istanza x' con $|x'| \leq |x|^c$ per una certa costante c. Dato che $A \in$ logAPX, è possibile trovare in tempo polinomiale una soluzione y $(k \log|x'|)$-approssimata per x', dove k' è una costante. Da y, tramite tecniche di ricerca localre, possiamo trovare una soluzione 2^{16}-approssimata per x' usando $\lceil \log \lceil \log_{2^{16}}(k \log|x'|) \rceil \rceil \leq \log \lceil \log\log |x|^{kc} \rceil - 3 \leq \log\log\log|x| - 2$ domande adattive ad NP, dove l'ultima disuguaglianza vale per $|x|$ sufficientemente grande. Perciò,

$$\mathsf{NPF}^{\mathsf{NP}[\log\log\log n - 1]} \subseteq \mathsf{NPF}^{\mathsf{NP}[\log\log\log n - 2]}$$

il che, secondo il Teorema 2.26, implica il collasso della gerarchia polinomiale. □

Abbiamo che se la gerarchia polinomiale non collasse, Min Set Cover e gli altri problemi logAPX-completi non sono self-improvable. Possiamo anche dimostrare che se NP = *co*-NP, allora Min Set Cover *è* self-improvable. La dimostrazione è molto simile a quella del Teorema 3.46 ed è omessa.

3.5.3 Soluzioni ottime e approssimate per Max Clique

Il lemma seguente è un analogo della Proposizione 3.34 all'interno di NPO PB. La dimostrazione è sostanzialmente identica.

Lemma 3.48 *Per ogni* NPO PB *problem A e for any $r > 1$,* $\mathsf{P}^{A_r} \subseteq \mathsf{P}^{\mathsf{NP}[\log\log n+O(1)]}$.

Da questo lemma, dal fatto che $\mathsf{P}^{\mathsf{NP}[\log n]}$ è contenuto in $\mathsf{P}^{\mathrm{MC}_1}$, dove MC sta per Max Clique, [82], e dal fatto che se esiste una costante k tale che

$$\mathsf{P}^{\mathsf{NP}[\log\log n+k]} = \mathsf{P}^{\mathsf{NP}[\log n]},$$

allora la gerarchia polinomiale collassa (vedi Teorema 2.26), segue il risultato seguente che risolve una questione aperta posta in [28]

Teorema 3.49 *Se* $\mathsf{P}^{\mathrm{MC}_1} \subseteq \mathsf{P}^{\mathrm{MC}_2}$ *allora la gerarchia polinomiale collassa.*

Capitolo 4

Risultati di non-approssimabilità

"Forty-two" said Deep Thought, with infinity majesty e calm.(...) "It was a tough assignment," said Deep Thought mildly.
"Forty-two!" yelled Loonquawl. " Is that all you've got to show for seven e a half million years' work?"
"I checked it very thoroughly," said the computer.
(D. Adams [2])

Questo capitolo introduce una serie di metodi generali per derivare migliore riduzioni approximation-preserving e quindi risultati di non-approssimabilità più forti. La nostra attenzione si concentra soprattutto sui *problemi di soddisfacimento di vincoli* (vedi la Sezione 2.1). Ricordiamo che, oltre a Max Sat e alle sue restrizioni/generalizzazioni, numeorsi altri problemi ammetono tale tipo di formulazione. Risultati di non-approssimabilità per questi problemi sono dimostrati usando riduzioni per sostituzione locale dal verificatore del teorema PCP. La nozione di "gadget", formalizzata da Bellare, Goldreich e Sudan [19], cattura il modo in cui tali riduzioni funzionano.

Introduciamo una tecnica generale che consente di trovare *automaticamente* (cioè attraverso un programma di computer) il gadget *migliore possibile* per ridurre una computazione di un verificatore a un problema dato di soddisfacimento di vincoli. Le migliori riduzioni trovate in questo modo danno nuovi risultati di non-approssimabilità per Max Cut, Max DiCut e Max 2CSP. Mostriamo anche come fare un uso *algoritmico* di gadget, e presentiamo nuovi algoritmi di approssimazione per Max 3Sat e Max 3CSP basati su riduzioni con gagdet a Max 2Sat.

I risultati di non-approssimabilità che dimostriamo (così come quelli per Max Sat, Max 2Sat, e altri problemi [19, 63]) valgono solo per le versioni pesate. Diamo una tecnica che estende tali risultati alle versioni non pesate.

Il problema Min Vertex Cover non sembra rientrare in questo contesto di problemi di soddisfacimento di vincoli. Dimostriamo invece che è equivalente a Min Sat, il problema di minimizzare il numero di clausole soddisfatte in una formula booleana CNF. Usando tecniche differenti, dimostriamo che Min Vertex Cover e Min Sat sono equivalenti alle loro versioni non pesate. Dimostriamo anche un risultato di non-approssimabilità per Min Vertex Cover su grafi con grado limitato che è essenzialmente tanto forte quanto il miglior risultato di non-approssimabilità per grafi arbitrari.

Le nostre riduzioni da problemi di soddisfacimento di vincoli pesati alle loro versioni non pesate e da Min Vertex Cover alla sua restrizione con grado limitato usano oggetti combinatori collegati a *grafi expander*. Costruzioni probabilistiche e deterministiche di tali oggetti sono presentate in una sezione separata.

4.1 Gadget e programmazione lineare

4.1.1 Definizioni

Definizione 4.1 (gadget) *Dati una costante* $\alpha \in \mathcal{R}^+$*, una funzione di vincolo* $f : \{0,1\}^k \to \{0,1\}$*, e una famiglia di vincoli* $\mathcal{F}$*, un* α-gadget *che riduce* f *a* $\mathcal{F}$ *è una collezione finita di pesi reali* $w_j \geq 0$*, e associati vincoli* C_j *da* $\mathcal{F}$ *su* variabili primarie $x_1, \ldots, x_k$ *e* variabili ausiliarie $y_1, \ldots, y_n$*, con la proprietà cje, per ogni assegnamento booleano* $\vec{a}$ *a* $x_1, \ldots, x_k$ *e* $\vec{b}$ *a* $y_1, \ldots, y_n$*, le seguenti condizioni sono soddisfatte:*

$$(\forall \vec{a} : f(\vec{a}) = 1)\ (\forall \vec{b}) : \quad \sum_j w_j C_j(\vec{a}, \vec{b}) \quad \leq \quad \alpha \tag{4.1}$$

$$(\forall \vec{a} : f(\vec{a}) = 1)\ (\exists \vec{b}) : \quad \sum_j w_j C_j(\vec{a}, \vec{b}) \quad = \quad \alpha, \tag{4.2}$$

$$(\forall \vec{a} : f(\vec{a}) = 0)\ (\forall \vec{b}) : \quad \sum_j w_j C_j(\vec{a}, \vec{b}) \quad \leq \quad \alpha - 1. \tag{4.3}$$

Il gadget è stretto *se, in aggiunta,*

$$(\forall \vec{a} : f(\vec{a}) = 0)\ (\exists \vec{b}) : \quad \sum_j w_j C_j(\vec{a}, \vec{b}) \quad = \quad \alpha - 1. \tag{4.4}$$

Diciamo che una funzione $\vec{b} = \vec{b}(\vec{a})$ *è un* testimone *per il gadget se l'equazione (4.2) (e, per gadget stretti, l'equazione (4.4))è soddisfatta da* $\vec{b}(\vec{a})$*. Data una funzione testimone* $\vec{b}$*, denotiamo* $b_i(\vec{a}) = \vec{b}(\vec{a})_i$*.*

Osserviamo che, dal momento che i pesi possono essere tutti scalati, quello che è siginificativo è il rapporto tra le parti destre delle equazioni (4.1–4.4). Un gadget "forte" è uno con un α piccolo; in particolare, se $\alpha' > \alpha$, ogni α-gadget è anche automaticamente (dopo una scalatura dei pesi) un α'-gadget (ma la proprietà di essere stretto non è mantenuta). All'estremo, un 1-gadget darebbe una riduzione perfetta.

Il nostro interesse nella costruzione di gadget, e nelle famiglie PC e RMBC (che, al contrario delle altre, non danno origine a problemi naturali di soddisfacimento di vincoli) è stato inizialmente dovuto al seguente risultato di Bellare, Goldreich e Sudan.

Lemma 4.2 *[19] Per ogni famiglia* $\mathcal{F}$*, se esistono* α*-gadget che riducano ogni elemento di* PC *a* $\mathcal{F}$ *e* β*-gadget che riducano ogni elemento di* RMBC *a* $\mathcal{F}$*, allora per ogni* $\gamma > 0$*, è* NP*-hard approssimare* Max $\mathcal{F}$ *entro*

$$1 + \frac{.15}{.6\alpha + .4\beta - .15} - \gamma\ .$$

Attualmente, il miglior punto di partenza per dimostrare risultati di non-approssimabilità è il seguente risultato di Håstad.

Lemma 4.3 *[63] Per ogni famiglia* $\mathcal{F}$*, se esiste un* α_0*-gadget che riduca* PC_0 *a* $\mathcal{F}$ *e un* α_1*-gadget che riduca* PC_1 *a* $\mathcal{F}$*, allora per ogni* $\gamma > 0$*, è* NP*-hard approssimare* Max $\mathcal{F}$ *entro*

$$1 + \frac{1}{\alpha_0 + \alpha_1 - 1} - \gamma\ .$$

Costruzioni di gadget che riducano RMBC a problemi naturali non hanno più applicazioni, e non saranno descritte in dettaglia, ma le menzioneremo comunque per completezza.

4.1.2 La procedura di base

In questa sezione illustreremo la nostra tecnica costruendo un gadget che riduca PC_0 a 2SAT.

Un aspetto fondamentale del nostro metodo è il limitare il numero di variabili ausiliare, dimostrando che duplicati (in un senso che sarà chiarificato tra poco) possono essere eliminati tramite opportune *sostituzioni*. In generale, ciò è possibile se la riduzione è verso una famiglia con una proprietà che chiamiamo in analogia con la terminologia per i matroidi.

Definizione 4.4 *Una famiglia di vincoli $\mathcal{F}$ è* ereditaria *se per ogni $f_i(x_1, \ldots, x_{n_i}) \in \mathcal{F}$, e per ogni coppia di indici distinti $j, j' \in [n_i]$, la funzione f_i, quando viene ristretta a $x_j \equiv x_{j'}$ e viene considerata come una funzione di $n_i - 1$ variabili, è identica (a meno dell'ordine degli argomenti) a qualche altra funzione $f_{i'} \in \mathcal{F} \cup \{0, 1\}$, dove $n_{i'} = n_i - 1$ (e 0 e 1 denotano le funzioni costanti).*

Lemma 4.5 *Se un α-gadget Γ che riduca f ad una famiglia ereditaria $\mathcal{F}$ ha un funzione testimone per la quale due variabili ausiliarie sono identica (cioè $b_{j'}(\cdot) \equiv b_j(\cdot)$), o se una variabile ausiliaria è identica ad una variabile primaria ($b_{j'}(\vec{a}) \equiv a_j$) allora esiste un α'-gadget Γ' (con $\alpha' \leq \alpha$) che usa una variabile ausiliare in meno. Se Γ è stretto, allora lo è anche Γ'.*

DIMOSTRAZIONE: Definiamo un nuovo gadget Γ' ottenuto da Γ rimpiazzando ogni occorrenza di $x_{j'}$ con x_j e mostriamo che Γ' è un α'-gadget che riduce f a $\mathcal{F}$ per qualche $\alpha' \leq \alpha$.

Per ogni vincolo C di Γ, definiamo $\mathrm{red}(C)$ nel modo seguente. Se $x_{j'}$ non occorre in C, allora semplicemente imponiamo $\mathrm{red}(C) = C$. Altrimenti, definiamo, come prima approssimazione, $\mathrm{red}(C)$ come il vincolo ottenuto da C rimpiazzando ogni occorenza di $x_{j'}$ con una occorrenza di x_j. Se C non conteneva originariamente x_j, allora $\mathrm{red}(C)$ è un vincolo valida da $\mathcal{F}$. Se invece C aveva già una occorrenza di x_j, allora $\mathrm{red}(C)$ ne ha due, che non è permesso dalla nostra definizione di vincolo. In questo caso, però, la ereditarietà di $\mathcal{F}$ ci assicura che $\mathrm{red}(C)$ è equivalente o a un altro vincolo C' da $\mathcal{F}$ oppure alla costante 0 o 1. In questo caso, imponiamo che $\mathrm{red}(C)$ sia uguale a C' o alla costante appropriata.

Se $ad = (C_1, \ldots, C_m, w_1, \ldots, w_m)$, allora definiamo un nuovo gadget $\Gamma' = (\mathrm{red}(C_1), \ldots, \mathrm{red}(C_m), w_1, \ldots, w_m)$. Corrispondentement, definiamo $\vec{b}'(\vec{a})$ come la funzione identica a $b(\vec{a})$ ma con $b_{j'}$ eliminato. Γ' ha effettivamente una variabile ausiliare in meno rispetto a Γ ($x_{j'}$ non occorre mai in Γ').

Per costruzione, $\Gamma'(\vec{a}, \vec{b}'(\vec{a})) \equiv \Gamma(\vec{a}, \vec{b}(\vec{a}))$, perciò Γ' soddisfa l'equatione (2) nella definizione di gadget (analogamente, se Γ era un gadget stretto, Γ' soddisfa (4)). Inoltre, la quantificazione universale nelle disuguaglianze (1) e (3) è più ristretta per Γ' che per Γ', perciò Γ' soddisfa sicuramente anche queste equazioni. Se Γ' contiene alcune costanti, sottraendole da entrambe le parti delle disuguaglianze (1–4) produce un $(\alpha - w)$-gadget, dove w è il peso totale dei vincoli che sono stati rimpiazzati dalla costante 1. $\square$

Se la famiglia di vincoli permette la complementazione di ogni variabile (come ad esempio 2SAT ma non CUT o DICUT), allora il numero di variabili ausiliarie può essere approssimativamente dimezzato: dobbiamo solo considerare funzioni testimone che valgono 1 almeno tanto spesso che 0.

Definizione 4.6 *Una famiglia $\mathcal{F}$ è* chiusa rispetto al complemento *se è ereditaria e, per ogni*

A family $\mathcal{F}$ è complementation-closed *if it è hereditary and, for $f_i(x_1, \ldots, x_{n_i}) \in \mathcal{F}$, e ogni coppia di indici distinti $j, j' \in [n_i]$, la funzione f_i, quando viene ristretta a $x_j \equiv \neg x_{j'}$ e viene vista* *come una funzione di $n_i - 1$ variabili, è identica (a meno dell'ordine degli argomenti) a qualche altra funzione $f_{i'} \in \mathcal{F} \cup \{0,1\}$, dove $n_{i'} = n_i - 1$ (e 0 e 1 denotano le funzioni costanti).*

La proprietà precedente garantisce che possiamo effetturare sostituzioni anche se due componenti della funzione testimone sono una il complemento dell'altra. Abbiamo bisogno di un'altra definizione prima di poter formulare il risultato principale di questa sezione.

Definizione 4.7 *Data una funzione di vincolo f, diremo che due variabili $a_{j'}$ e a_j sono* distinte *se esiste un assegnamento $\vec{a}$ che soddisfa f e per il quale $a_{j'} \neq a_j$.*

Lemma 4.8 *Sia f una funzione di vincolo su k variabili, sia s il numero di assegnamenti che soddisfano f, e si assume che f abbia $k' \leq k$ variabili distinte anche rispetto al complemento. Se c'è un α-gadget che riduca f ad una famiglia di vincoli $\mathcal{F}$ ereditaria (rispettivamente, chiusa rispetto al complemento) allora esiste un α'-gadget (con $\alpha' \leq \alpha$ con al più $2^s - k'$ (rispettivamente, $2^{s-1} - k'$) variabili ausiliarie. Se esiste un α-gadget stretto che riduca f ad una famiglia di vincoli $\mathcal{F}$ ereditaria (rispettivamente, chiusa rispetto al complemento), allora esiste un α'-gadget (con $\alpha' \leq \alpha$) stretto con al più $2^k - k'$ (rispettivamente, $2^{k-1} - k'$) variabili ausiliarie.*

In alcuni casi (ad esempio 2SAT ma non CUT, non è nemmeno necessario considerare funzioni testimone che sono identicamente 0 o identicamente 1.

Come conseguenza della discussione precedente, abbiamo che, se stiamo cercando un α-gadget che riduca PC_0 a 2SAT con il minimo valore di α, allora possiamo restringere la nostra ricerca a gadget con al più 7 variabili. Su 7 variabili è possibile definire $2 \cdot 7 + 4 \cdot \binom{7}{2}$ vincoli di 2SAT; li chiameremo $C_1, \ldots, C_{98}$. Un gadget su 7 variabili può dunque essere identificato con il vettore $(w_1, \ldots, w_{98})$ dei pesi dei vincoli. In [19] è stato mostrato che esiste un 11-gadget che riduce PC_0 a 2SAT, ne consegue che in un gadget ottimo nessun vincolo avrà un peso maggiore di 11. Se ci limitassimo a considerare gadget con pesi interi, allora un gadget ottimo potrebbe essere trovato tramite una ricerca esaustiva su uno spazio di 12^{98} elementi. Se invece ammettiamo pesi reali sui vincoli, lo spazio di ricerca diventa infinito, ma potremo usare programmazione lineare per trovare un gadget ottimo con una certa efficienza. Si consideri il seguente programma lineare con $64 + 4 + 64 = 132$ vincoli su 99 variabili:

$$
\begin{array}{lll}
\text{minimize} & \alpha & \\
\text{subject to} & & \\
& \sum_j w_j C_j(\vec{a}, \vec{b}) \leq \alpha & (\forall \vec{a} : \mathrm{PC}_0(\vec{a}) = 1)\ (\forall \vec{b} \in \{0,1\}^4) \\
& \sum_j w_j C_j(\vec{a}, b(\vec{a})) = \alpha & (\forall \vec{a} : \mathrm{PC}_0(\vec{a}) = 1) \\
& \sum_j w_j C_j(\vec{a}, \vec{b}) \leq \alpha - 1 & (\forall \vec{a} : \mathrm{PC}_0(\vec{a}) = 0)\ (\forall \vec{b} \in \{0,1\}^4) \\
& \alpha \geq 0 & \\
& w_j \geq 0 & (\forall j \in [98])
\end{array}
\tag{PL1}
$$

In generale, tutte le variabili "duplicate" possono essere eliminate da un α-gadget per dare un α'-gadget ($\alpha' \leq \alpha$) "semplificato", e l'α'-gadget può essere associato con il suo vettore di pesi, di lunghezza fissata.

Teorema 4.9 *Il vettore dei pesi associato con una α-gadget semplificato è una soluzione ammissibile per l'associato PL, e ogni soluzione ammissibile per il PL è un vettore di pesi che descrive un α-gadget. Una soluzione ottime per il PL descrive dunque un α-gadget ottimale (cioè in cui α assume il valore più piccolo possibile).*

In particolare, (PL1) ha come soluzione ottima $\alpha = 11$, e ciò dimostra l'ottimalità del gadget di [19].

4.2 Risultati di non-approssimabilità usando gadget

In questa sezione troveremo gadget che riducano elementi di PC a CUT, DICUT, e 2CSP. I nostri gadget, insieme al Lemma 4.3 implicano i migliori risultati di non-approssimabilità noti per tali problemi.

4.2.1 Il problema MAX CUT

Cominciamo con il mostrare un nuovo risultato di non-approssimabilità per il problema MAX WEIGHT CUT. Non è difficile vedere che nessun gadget che soddisfi la Definizione 4.1 può ridurre PC_0 a CUT. Per ogni assegnamento delle variabili che soddisfi l'equazionen (4.2), l'assegnamento complementare ha la parità opposta (e dunque dovrebbe essere soggetto alla disuguaglianza (4.3)), ma il valore di tutti i vincoli CUT non è cambiato (perciò il valore del gadget è sempre α violando (4.3)). Seguendo [19], generalizziamo la definizione di gadget.

Definizione 4.10 *Un* gadget con costante ausiliaria 0 *è un gadget come definito precedentemente, con l'eccezione che (4.1–4.4) valgono solo quando $y_1 = 0$.*

Per ottenere un risultato di non-approssimabilità per MAX CUT, abbiamo prima bisogno del seguente lemma, che è una piccola modifica del Lemma 4.3.

Lemma 4.11 *[63] Se esiste un α_0-gadget con costante 0 che riduca PC_0 a CUT e un α_1-gadget con costante 0 che riduca PC_1 a CUT, allora per ogni $\gamma > 0$, è NP-hard approssimare MAX CUT entro*

$$1 + \frac{1}{\alpha_1 + \alpha_2 - 1} - \gamma \; .$$

Si noti che la famiglia di vincoli CUT è ereditaria, dato che identificare due variabili in un vincolo CUT crea la funzione costante 0. Perciò, in base al Lemma 4.8, se esiste un α-gadget con costante 0 che riduca un elemento di PC a CUT, allora esiste un α-gadget con al più 13 variabili ausiliarie (16 variabili in tutto). Sono possibili solo $\binom{16}{2} = 120$ vincoli CUT su 16 variabili. Dato che dobbiamo considerare solo i casi in cui $y_1 = 0$, possiamo costruire un programma lineare come prima con $2^{16-1} + 4 = 32772$ vincoli per trovare gli α-gadget con costante 0 ottimi che riducano gli elementi di PC a CUT.

Lemma 4.12 *Esiste un 8-gadget con costante 0 che riduce PC_0 a CUT, ed è ottimo e stretto.*

Mostriamo il gadget risultante nella Figura 4.1 come un grafo. Ogni vincoli il cui peso sia diverso da zero è mostrato come un arco. Si noti che alcune variabili ausiliarie non appaiono in alcun vincolo di peso positivo, e sono perciò omesse dal grafo. Tutti i vincoli di peso non nullo hanno peso 1/2. La variabile ausiliare $y_1 = 0$ è mostrata come il nodo "0" nel grafo.

Lemma 4.13 *Esiste un 9-gadget con costante 0 che riduce PC_1 a CUT, ed è ottimo e stretto.*

I due lemmi implicano il seguente teorema.

Teorema 4.14 *Per ogni $\gamma > 0$, è NP-hard approssimare MAX CUT entro $17/16 - \gamma$.*

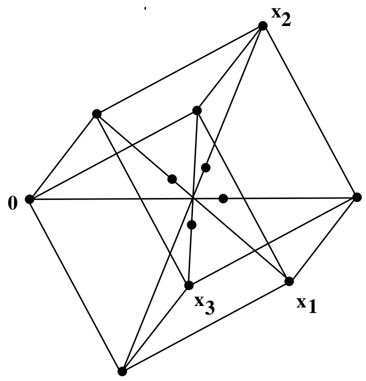


Figura 4.1: L' 8-gadget che riduce PC_0 a CUT. Ogni arco ha peso $0,5$, La variabile ausiliare che vale sempre 0 è etichettata con 0.

I gadget RMBC Trovare gadget per RMBC è stato più difficile. Discutiamo questo punto perché ci ha portati ad un'idea che potrebbe trovare applicazione in altre riduzioni. È subito risultato chiare che non fosse posibile applicare esattamente le tecniche precedenti per trovare un gadget ottimo che riducesse, ad esempio, $RMBC_{00}$ a CUT. Essendoci 8 assegnamenti che soddisfano le 4 variabili del vincolo $RMBC_{00}$, il Lemma 4.8 ci dice che dobbiamo considerare $2^8 - 4 = 252$ variabili ausiliarie, e quindi scrivere un programma lineare con $2^{252} + 8$ vincoli, che è un pò al di là delle capacità degli attuali risolutori di programmi lineari. Per superare questa difficoltà, abbiamo osservato che, per la funzione $RMBC_{00}$, quando $a_1 = 0$, il valore di a_4 è irrilevante, e quando $a_1 = 1$,il valore di a_3 è irrilevante. Perciò abbiamo considerato solo funzioni testimone ristrette per le quali $\vec{b}(0, a_2, a_3, 0) = \vec{b}(0, a_2, a_3, 1)$ e $\vec{b}(1, a_2, 0, a_4) = \vec{b}(1, a_2, 1, a_4)$. A priori, non era ovvio che esistesse un gadget con una tale funzione testimone, ma abbiamo assunto per il momento che tale gadget esistesse. Seguendo la dimostrazione del Lemma 4.8, abbiamo notato che la taglia del dominio della funzione testimone ristretta è diventata 4 (anziché 8). Quindi, se esistesse un α-gadget con costante 0 e con funzione testimone ristretta, esisterebbe anche un α-gadget con costante 0 e al più $2^4 - 2 = 14$ variabili ausiliarie. A questo punto, abbiamo creato un programma lineare con $\binom{18}{2} = 153$ variabili e $2^{18-1} + 8 = 131080$ vincoli. Il risultato del programma lineare è stato che esiste un 8-gadget con costante 0 che riduce $RMBC_{00}$ a CUT, e che tale gadget è stretto. Si noti che, avendo usato una funzione testimone ristretta, il programma lineare non dimostra che il gadget in questione sia ottimo. Comunque, la sua ottimalità può essere dimostrata, e ciò sarà discusso nella Sezione 4.4.

4.2.2 Il problema MAX DICUT

Nel caso dei vincoli DICUT, non abbiamo bisogno di gadget con costante. Usando argomentazioni simili a quelle precedenti, si vede che se esiste un α-gadget che riduce un elemento di PC a DICUT allora esiste un α-gadget con 13 variabili ausiliarie. Questo ci porta a formulare programmi lineari con $16 \cdot 15$ variabili (una per ogni possibile vincolo DICUT, che corrisponde ad un arco diretto) e $2^{16} + 4 = 65540$ vincoli. La soluzione dei programmi lineari è data nei seguenti lemmi.

Lemma 4.15 *Esiste un* 6.5*-gadget che riduce* PC_0 *a* DICUT*, ed è ottimo e stretto.*

Mostriamo il gadget risultato nella Figura 4.2.

Lemma 4.16 *Esiste un* 6.5*-gadget che riduce* PC_0 *a* DICUT*, ed è ottimo e stretto.*

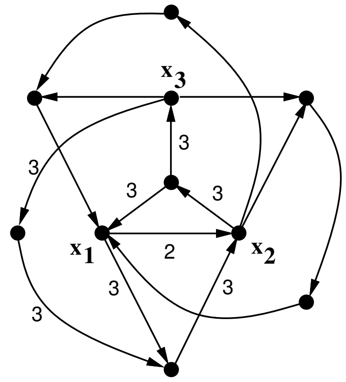


Figura 4.2: Il 6.5-gadget che riduce PC_0 a DiCut. Gli archi hanno peso 1 eccetto quando indicato diversamente.

I due lemmi implicano il seguente teorema.

Teorema 4.17 *Per ogni* $\gamma > 0$*,* Max DiCut *è* NP*-hard da approssimare entro* $13/12 - \gamma$*.*

I gadget RMBC Usando l'idea di restringere la funzione testimone come nella riduzione da RMBC a Max Cut, anche in questo caso abbiamo creato il programma lineare che trova α-gadget che riducano RMBC a DiCut con funzione testimone limitata. Il programma lineare aveva $18 \cdot 17 = 306$ variabili e $2^{18} + 8 = 262152$ vincoli. Avendo considerato solo funzioni testimone ristrette, il programma lineare non produce necessariamente un gadget ottimo, ma dimostriamo nella Sezione 4.4 che non può esistere alcun gadget migliore. I nostri risultati finali sono stati che esiste un 6-gadget che riduce $RMBC_{00}$ a DiCut, ed è stretto; esistono 6.5-gadget che riducono $RMBC_{01}$ e $RMBC_{10}$ a DiCut, e sono stretti; esiste un 7-gadget che riduce $RMBC_{11}$ a DiCut, ed è stretto.

4.2.3 Il problema Max 2CSP

Nel caso della famiglia 2CSP, così come per 2SAT, dobbiamo solo considerare 4 variabili ausiliarie, o 7 variabili in totale. Ci sono due funzioni non costanti su una sola variabile, e 12 funzioni su una coppia di variabili distinte, perciò dobbiamo considerare $2 \cdot 7 + 12 \cdot \binom{7}{2} = 266$ funzioni in totale. I programmi lineari corrispondenti hanno una variabile per ogni funzione (più una per α) e $2^7 + 4 = 132$ vincoli. Le soluzioni ottime danno il seguente lemma.

Lemma 4.18 *Esistono* 5*-gadget che riducono* PC_0 *e* PC_1 *a* 2CSP*, e sono ottimi e stretti.*

Il gadget che riduce PC_0 a 2CSP è il seguente:

$$\begin{array}{cccc} x_1 \wedge \neg y_1, & x_1 \wedge y_2, & \neg x_1 \wedge y_3, & \neg x_1 \wedge y_4, \\ x_2 \wedge \neg y_1, & \neg x_2 \wedge y_2, & x_2 \wedge y_3, & \neg x_2 \wedge y_4, \\ \neg x_3 \wedge y_1, & x_3 \wedge \neg y_2, & x_3 \wedge \neg y_3, & \neg x_3 \wedge \neg y_4. \end{array}$$

Il gadget che riduce PC_1 a 2CSP può essere ottenuto da quello precedente complementando tutte le occorrenze di x_1.

Teorema 4.19 *Per ogni* $\gamma > 0$*,* Max 2CSP *è* NP*-hard da approssimare entro* $10/9 - \gamma$*.*

Max Weight 2CSP può essere approssimato entro 1.169 [45]. Il teorema precedente ha implicazioni per le dimostrazioni verificabili probabilisticamente. usando la nota riduzione da problemi di soddisfacimento di vincoli a dimostrazioni verificabili probabilisticamente, il Teorema 4.19 implica il seguente teorema.

Teorema 4.20 *Per ogni* $\epsilon > 0$*, esistono costanti* c *ed* s *tali che* $\mathsf{NP} \subseteq \mathsf{PCP}_{c,s}[\log, 2]$ *e* $c/s > 10/9 - \epsilon$*.*

I gadget RMBC Il risultato per i gadget RMBCè stato il seguent: [er ogni elemento di RMBC, esiste un 5-gadget che lo riduce a 2CSP, ed è ottimo e stretto.

4.3 Algoritmi di approssimazione usando gadget

In questa sezione mostriamo come fare uso di gadget per migliorare algoritmi di approssimazione. In particolare, considereremo i problemi Max Weight 3SAT e Max Weight 3ConjSAT. Un miglior algoritmo di approssimazione per il secondo problema ha come implicazione l'allargamento della famiglia di classi PCP note essere in P. Entrambi i nostri nuovi algoritmi di approssimazione si basano sul ridurre il problema a Max Weight 2SAT usando gadget. Cominciamo con l'introdurre un po' di notazione.

Definizione 4.21 *Un algoritmo* (β_1, β_2)*-approssimato per* Max Weight 2SAT *è un algoritmo che riceve in input una istanza con clausole unarie di peso totale* m_1 *e clausole binarie di peso totale* m_2*, e due reali* u_1 *e* u_2*, e produce due reali* $s_1 \leq u_1$ *e* $s_2 \leq u_2$ *e un assegnamento che soddisfa clausole di peso totale almeno* $\beta_1 s_1 + \beta_2 s_2$*. Se esiste una soluzione ottima che soddisfa clausole unarie di peso totale al più* u_1 *e clausole binarie di peso totale al più* u_2*, allora c'è la garanzia che nessun assegnamento soddisfa clausole di peso totale maggiore di* $s_1 + s_2$*.*

In altre parole, un algoritmo (β_1, β_2)-approssimato, data una coppia di "limitazioni superiori" per l'ottimo u_1, u_2, produce una singola limitazione superiore $s_1 + s_2$, assieme ad un assegnamento che dimostra una limitazione inferiore $\beta_1 s_1 + \beta_2 s_2$.

Lemma 4.22 *[45] Esiste un algoritmo polinomiale* $(.976, .931)$*-approssimato per* Max Weight 2SAT*.*

4.3.1 Il problema Max 3Sat

In questa sezione mostriamo come derivare un miglior algoritmo di approssimazione per Max Weight 3SAT. Restringendo le tecniche di [58] da Max Weight Sat a Max Weight 3SAT e usando un algoritmo $1,074$-approssimato per Max Weight 2SAT dovuto a Feige e Goemans [45], è possibile ottenere un algoritmo $1,299$-approssimato per Max Weight 3SAT. L'idea di base di [58] è di ridurre ogni clausola di lunghezza 3 alle tre possibili sottoclausole di lunghezza 2, di dare a ciascuna di queste nuove clausole di lunghezza 2 un terzo del peso originale, e poi applicare l'algoritmo di approssimazione per Max Weight 2SAT. Questo algoritmo di approssimazione è poi "bilanciato" con un altri algoritmo di approssimazione per Max Weight 3SAT per ottenere il risultato finale. Nel seguito mostriamo che usando un gadget stretto per ridurre 3SAT a Max Weight 2SAT, un buon algoritmo (β_1, β_2)-approssimato consente di ottenere un algoritmo $1,249$-approssimato per Max Weight 3SAT.

Lemma 4.23 *Se, per ogni $f \in$ E3SAT, esiste un α-gadget stretto che riduce f a 2SAT, esiste un algoritmo (β_1, β_2)-approssimato per* Max Weight 2SAT*, e $\alpha \geq \frac{(\beta_1-\beta_2)}{2(1-\beta_2)} + 1$, allora esiste un algoritmo $(1/\rho)$-approssimato per* Max Weight 3SAT *con*

$$\rho = \frac{1}{2} + \frac{(\beta_1 - 1/2)(3/8)}{(\alpha-1)(1-\beta_2) + (\beta_1 - \beta_2) + (3/8)}.$$

Dimostrazione: Sia ϕ una istanza di Max Weight 3SAT con clausole di unarie di peso totale m_1, clausole binarie di peso totale m_2, e clausole ternarie di peso totale m_3. Usiamo i due algoritmi elencati nel seguito, ottenendo i corrispondenti limiti superiori e inferiori al numero di clausole soddisfacibili.

- Random: diamo ad ogni variabile valore 1 con probabilità 1/2. Questo da una soluzione di peso almeno $m_1/2 + 3m_2/4 + 7m_3/8$.
- Programmazione semidefinita: usiamo gli α-gadget stretti per ridurre ogni clausola di lunghezza 3 a clausole di lunghezza 3. Questo fornisce una istanza di Max Weight 2SAT, alla quale applichiamo l'algoritmo (β_1, β_2)-approssimato con parametri $u_1 = m_1$ e $u_2 = m_2 + \alpha m_3$. L'algoritmo di approssimazione restituisce un limite superiore $s_1 + s_2$ sul peso totale di ogni soluzione per l'istanza di Max Weight 2SAT e un assegnamento di peso $\beta_1 s_1 + \beta_2 s_2$. Quando viene ricondotto all'istanza di Max Weight 3SAT, l'assegnamento ha peso almeno $\beta_1 s_1 + \beta_2 s_2 - (\alpha-1)m_3$. Inoltre, $s_1 \leq m_1$, $s_2 \leq m_2 + \alpha m_3$, e l'ottimo della istanza di Max Weight 3SAT è al più $s_1 + s_2 - (\alpha - 1)m_3$.

Il rapporto di prestazione dell'algoritmo che prende la migliore delle due soluzioni precedenti è almeno $(1/\rho_1)$, dove

$$\rho_1 \stackrel{\text{def}}{=} \min_{\substack{s_1 \leq m_1 \\ s_2 \leq m_2 + \alpha m_3}} \frac{\max\{m_1/2 + 3m_2/4 + 7m_3/8, \quad \beta_1 s_1 + \beta_2 s_2 - (\alpha-1)m_3\}}{s_1 + s_2 - (\alpha-1)m_3}$$

Definiamo ora una serie di semplificazioni che ci aiuteranno a provare la limitazione richiesta dal lemma.

$$\rho_2 \stackrel{\text{def}}{=} \min_{\substack{t_1 \leq m_1 \\ t_2 \leq m_2 + m_3}} \frac{1}{t_1 + t_2} \max\{ \begin{array}{l} m_1/2 + 3m_2/4 + 7m_3/8, \\ \beta_1 t_1 + \beta_2 t_2 - (1-\beta_2)(\alpha-1)m_3\} \end{array}$$

$$\rho_3 \stackrel{\text{def}}{=} \min_{\substack{t_1 \leq m_1 \\ t_2 \leq m_2 + m_3}} \frac{1}{t_1 + t_2} \max\{ \begin{array}{l} t_1/2 + 3t_2/4 + m_3/8, \\ t_1/2 + 7m_3/8, \\ \beta_1 t_1 + \beta_2 t_2 - (1-\beta_2)(\alpha-1)m_3\} \end{array}$$

$$\rho_4 \stackrel{\text{def}}{=} \min_{t_2 \leq t} \frac{1}{t} \max\{ \begin{array}{l} t/2 + t_2/4 + m_3/8, \\ t/2 - t_2/2 + 7m_3/8, \\ \beta_1 t - (\beta_1 - \beta_2)t_2 - (1-\beta_2)(\alpha-1)m_3\} \end{array}$$

$$\rho_5 \stackrel{\text{def}}{=} \frac{1}{2} + \left(\frac{\frac{3}{8}(\beta_1 - \frac{1}{2})}{(1-\beta_2)(\alpha-1) + (\beta_1 - \beta_2) + \frac{3}{8}} \right)$$

Per finire la prova del lemma, ci rimane da convincerci del fatto che

$$\rho_1 \geq \rho_2 \geq \cdots \geq \rho_5.$$

La prima disuguaglianza segue dalla sostituzione di variabili $t_1 = s_1$, $t_2 = s_2-(\alpha-1)m_3$. La seconda segue dal fatto che imponendo $m_1 = t_1$ e $m_2 = \max\{0, t_2-m_3\}$, possiamo solo ridurre il numeratore. La terza disuguaglianza si ottiene imponendo $t = t_1 + t_2$. La quarta segue sostituendo l'operatore max con una combinazione convessa dei suoi argomenti e poi semplificando. La combinazione convessa prende una frazione θ_1 del primo argomento, θ_2 del secondo e θ_3 del terzo, dove

$$\theta_1 = \frac{\frac{2}{3}(1-\beta_2)(\alpha-1)+\frac{7}{6}(\beta_1-\beta_2)}{(1-\beta_2)(\alpha-1)+(\beta_1-\beta_2)+\frac{3}{8}},$$

$$\theta_2 = \frac{\frac{1}{3}(1-\beta_2)(\alpha-1)-\frac{1}{6}(\beta_1-\beta_2)}{(1-\beta_2)(\alpha-1)+(\beta_1-\beta_2)+\frac{3}{8}}$$

and

$$\theta_3 = \frac{\frac{3}{8}}{(1-\beta_2)(\alpha-1)+(\beta_1-\beta_2)+\frac{3}{8}}.$$

Si osservi che $\theta_1 + \theta_2 + \theta_3 = 1$ e che la condizione su α garantisce che $\theta_2 \geq 0$. □

Osservazione 4.24 *L'analisi fornita nella dimostrazione del lemma precedente è la migliore possibile. In particolare, per una istanza con* m *clausole tale che*

$$m_3 \stackrel{\text{def}}{=} m\frac{\beta_1 - 1/2}{(1-\beta_2)(\alpha-1)+(\beta_1-\beta_2)+3/8}\ ,$$

$m_1 \stackrel{\text{def}}{=} m - m_3$, $m_2 \stackrel{\text{def}}{=} 0$, $s_1 = m_1$, *e* $s_2 = \alpha m_3$, *è facile vedere che* $\rho_1 = \rho_5$.

Lemma 4.25 *Per ogni funzione* $f \in$ E3SAT*, esiste un* 3.5*-gadget stretto che riduce* f *a* 2SAT.

DIMOSTRAZIONE: Il gadget è

$$x_1 \vee x_3, \quad \neg x_1 \vee \neg x_3,$$
$$x_1 \vee \neg y, \quad \neg x_1 \vee y,$$
$$x_3 \vee \neg y, \quad \neg x_3 \vee y,$$
$$x_2 \vee y,$$

Dove ogni clausole eccetto l'ultima ha peso 1/2; l'ultima clausola ha peso 1. □

Combinando i Lemmi reflm:fego, 4.23 e4.25 otteniamo un algoritmo 1.249-approssimato.

Teorema 4.26 *Esiste un algoritmo polinomiale* 1.249*-approssimato per* MAX WEIGHT 3SAT.

4.3.2 Il problema Max 3CSP

Passiamo adesso al problema Max Weight 3CSP. Per approssimare tale problema è sufficiente approssimare la sua versione ristretta Max Weight 3ConjSAT. Ciò è una conseguenza di un risultato più generale, non ristretto a vincoli ternari.

Teorema 4.27 *Per ogni $k > 1$ e per ogni $r \geq 1$, se* Max Weight kConjSAT *è r-approssimabile, allora* Max Weight k-CSP *è r-approssimabile.*

Dimostrazione: Senza perdere in generalità, possiamo assumere che l'arità di ogni vincolo sia esattamente k (infatti, ogni funzione h-aria, con $h < k$ può essere vista come una funzione k-aria il cui valore è indipendente da $k - h$ dei suoi argomenti). Sia $(C_1, \ldots, C_m, w_1, \ldots, w_m)$ una istanza di Max Weight k-CSP su variabili booleane $x_1, \ldots, x_n$. Per ogni $j \in [m]$, sia $C_j = f_j(x_{j_1}, \ldots, x_{j_k})$ dove $j_1, \ldots, j_k \in [n]$ e $f_j : \{\mathsf{true}, \mathsf{false}\}^k \to \{\mathsf{true}, \mathsf{false}\}$. Si consideri ora l'insieme $S_{f_j} \subseteq \{\mathsf{true}, \mathsf{false}\}^k$ degli assegnamenti che soddisfano f_j. Per ognuno di tali assegnamenti $b_1, \ldots, b_k$ possiamo scrivere un clausola k-aria congiuntiva che è soddisfatta solamente da quell'assegnamento. La clausola in questione sarà $l_{j_1} \wedge \ldots \wedge l_{j_k}$ dove l_{j_i} sta per x_{j_i} se $b_i = \mathsf{true}$, e per $\neg x_{j_i}$ altrimenti. In questo modo, possiamo convertire ogni vincolo C_j in $|S_{f_j}|$ nuovi vincoli congiuntivi. Diamo peso w_j a tali vincoli. Sia $D_1, \ldots, D_{m'}$ l'unione di questi nuovi insiemi di vincoli. Essi costituiscono una istanza di Max Weight kConjSAT su variabili $x_1, \ldots, x_n$, e dovrebbe essere chiaro che ogni assegnamento a $x_1, \ldots, x_n$ ha la stessa misura tanto per $C_1, \ldots, C_m$ che per $D_1, \ldots, D_{m'}$, così che le due istanze hanno lo stesso ottimo ed una soluzione r-approssimata per $D_1, \ldots, D_{m'}$ è anche r-approssimata per $C_1, \ldots, C_m$. $\square$

Algoritmi di approssimazione per Max kConjSAT implicano anche il contenimento in P di alcune classi PCP.

Teorema 4.28 (Max kConjSAT e PCP) *Se* Max kConjSAT *è r-approssimabile per qualche $r \geq 1$, allora $\mathsf{PCP}_{c,s}[\log, k] \subseteq \mathsf{P}$ per ogni $c/s > r$.*

Dimostrazione: Sia $L \in \mathsf{PCP}_{c,s}[\log, k]$ e e sia V il corrispondente verificatore. Sia x una stringa di lunghezza n. Sia l la massima lunghezza di una dimostrazione per x (si noti che l è polinomiale in l). Ricevendo x in input, V accede a $\alpha \log n$ bit random (dove α è una costante che dipende solo da V) e legge q bit della prova. Mostriamo adesso che per ognuna delle sue computazioni deterministiche esiste un 1-gadget che la riduce a kConjSAT.

Sia $\mathsf{r} = 2^{\alpha \log n}$ il numero totale (polinomiale in n) di possibili sequenze di but random a cui V accede. Per ognuna di tali stringhe $R \in \{0,1\}^{\alpha \log n}$, consideriamo il comportamento di V con input x e stringa random R: chiaramente tale comportamento è del tutto determinato dall'esito dei suoi accessi alla prova, e può essere rappresentato come un albero binario di altezza k In tale albero, ogni vertice interno corrisponde alla lettura di un bit, una diramazione corrisponde ai possibili esiti della lettura, e le foglie corrispondono agli stati finali che possono assere accettanti o rigettanti. Ogni cammino radice-foglia corrisponde ad un insieme di letture (posizioni sulla prova) $q_1, \ldots, q_k$ e a risposte $a_1, \ldots, a_k$, ed è seguito dal verificatore se e solo se $\pi[q_i] = a_i$ per ogni $i \in [k]$. Associamo adesso una variabile booleana x_i ad ogni posizione $\pi[i]$ della prova, per ogni $i \in [l]$. C'è una corrispondenza immediata tra prove (cioè stringhe in $\{0,1\}^l$) e assegnamenti per tali variabili In accordo con questa intuizione, codifichiamo i cammini radice-foglia che portano a stati accettanti tramite clausole congiuntive nel modo naturale: un cammino che corrisponda a letture $q_1, \ldots, q_k$ con esito $a_1, \ldots, a_k$ è convertito nella clausola congiuntiva $l_{q_1} \wedge \ldots, l_{q_k}$ dove l_{q_i} sta per x_{q_i} se $a_i = 1$, e sta per $\neg x_{q_i}$ altrimenti.

Dovrebbe essere chiaro che queste clausole sono tutte inconsistenti una con l'altra e che un assegnamento ne soddisfa una se e solo se la corrispondente prova fa accettare il verificatore, quando quest'ultimo usa la stringa random R. Possiamo ripetere questa costruzione per ogni R, e ciò porta a creare una istanza di MAX kSAT tale che i clausole sono simulataneamente soddisfacibili se e solo se esiste una prova che faccia accettare V con probabilità i/r. Ne consegue che l'ottimo di questa istanza di MAX kCONJSAT è almeno $c\mathsf{r}$ se $x \in L$, e più $s\mathsf{r}$ altrimenti. Ogni algoritmo di approssimazione con rapporto di prestazione migliore di c/s è sufficiente per distinguere i due casi. $\square$

Presenteremo adesso un algoritmo di approssimazione per MAX WEIGHT 3CONJSAT. La nostra analisi sarà simile a quella della sezione precedente.

Lemma 4.29 *Se per ogni $f \in$ 3CONJSAT esiste un $(\alpha_1 + \alpha_2)$-gadget stretto che riduce f a 2SAT composto da α_1 clausole unarie e da α_2 clausole binarie, ed esiste un algoritmo (β_1, β_2)-approssimato per MAX WEIGHT 2SAT, allora esiste un algoritmo $(1/\rho)$-approssimato per MAX WEIGHT 3CONJSAT con*

$$\rho = \frac{\frac{1}{8}\beta_1}{\frac{1}{8} + (1-\alpha_1)(\beta_1 - \beta_2) + (1-\beta_2)(\alpha_1 + \alpha_2 - 1)}$$

purché $\alpha_1 + \alpha_2 > 1 + 1/8(1 - \beta_2)$.

DIMOSTRAZIONE: Sia ϕ una istanza di MAX WEIGHT 3CONJSAT con vincoli di peso totale m. Come nel caso di MAX WEIGHT 3SAT, usiamo due algoritmi e prendiamo la migliore delle due soluzioni.

- Random: per ogni variabile, le assegnamo valore 1 con probabilità 1/2. Il peso totale dei vincoli soddisfatti è almeno $m/8$.
- Programmazione seimidefinita: usiamo l'α-gadget stretto per ridurre ogni vincolo a clausole di 2SAT. Questo fornisce una istanza di MAX WEIGHT 2SAT e usiamo l'algoritmo (β_1, β_2)-approssimato con parametri $u_1 = \alpha_1 m$ e $u_2 = \alpha_2 m$. L'algoritmo restituisce un limite superiore $s_1 + s_2$ sul peso totale dei vincoli soddisfatti nell'istanza di MAX WEIGHT 2SAT e un assegnamento di misura almeno $\beta_1 s_1 + \beta_2 s_2$. Quando viene ricondotto ad un assegnamento per l'istanza di MAX WEIGHT 3CONJSAT, la sua misura è almeno $\beta_1 s_1 + \beta_2 s_2 - (\alpha_1 + \alpha_2 - 1)m$. Inoltre, $s_1 \le \alpha_1 m$, $s_2 \le \alpha_2 m$, e il peso totale dei vincoli soddisfacibili nell'istanza MAX WEIGHT 3CONJSAT è al più $s_1 + s_2 - (\alpha_1 + \alpha_2 - 1)m$.

Abbiamo dunque che il rapporto di prestazione dell'algoritmo che prende la migliore delle due soluzioni è più $1/\rho_1$ dove ρ_1 è definito nel modo seguente.

$$\rho_1 \stackrel{\text{def}}{=} \min_{\substack{s_1 \le \alpha_1 m \\ s_2 \le \alpha_2 m}} \frac{\max\{m/8, \ \ \beta_1 s_1 + \beta_2 s_2 - (\alpha_1 + \alpha_2 - 1)m\}}{s_1 + s_2 - (\alpha_1 + \alpha_2 - 1)m}$$

Definiamo adesso una sequenza di semplificazioni.

$$\rho_2 \stackrel{\text{def}}{=} \min_{\substack{t_1 \le \alpha_1 m \\ t_2 \le (1-\alpha_1)m}} \frac{1}{t_1 + t_2} \max\{m/8, \ \ \beta_1 t_1 + \beta_2 t_2 - (1-\beta_2)(\alpha_1 + \alpha_2 - 1)m\}$$

$$\rho_3 \stackrel{\text{def}}{=} \min_{\substack{t \le m \\ t_2 \le (1-\alpha_1)m}} \frac{1}{t} \max\{m/8, \quad \beta_1 t - (\beta_1 - \beta_2)t_2 - (1-\beta_2)(\alpha_1 + \alpha_2 - 1)m\}$$

$$\rho_4 \stackrel{\text{def}}{=} \min_{t \le m} \frac{1}{t} \max\{m/8, \quad \beta_1 t - ((1-\alpha_1)(\beta_1 - \beta_2) + (1-\beta_2)(\alpha_1 + \alpha_2 - 1))m\}$$

$$\rho_5 \stackrel{\text{def}}{=} \frac{\frac{1}{8}\beta_1}{\frac{1}{8} + (1-\alpha_1)(\beta_1 - \beta_2) + (1-\beta_2)(\alpha_1 + \alpha_2 - 1)}$$

Per dimostrare il lemma, ci rimane da verificare che

$$\rho_1 \ge \rho_2 \ge \cdots \ge \rho_5.$$

La prima disuguaglianza segue dalla sostituzione di variabili $t_1 = s_1$ e $t_2 = s_2 - (\alpha_1 + \alpha_2 - 1)m$. La seconda segue dalla sostituzione $t = t_1 + t_2$. La terza disuguaglianza segue dal fatto che fissando t_2 a $(1-\alpha_1)m$ possiamo solo diminuire il numeratore. La quarta si ottiene sostituendo una combinazione convessa di argomenti al posto di max e poi semplificando. □

Lemma 4.30 *Per ogni $f \in$ 3ConjSAT esiste un 4-gadget che riduce f a 2SAT. Il gadget è composto di una clausola unaria e di tre clausole binarie.*

Dimostrazione: Dal momento che 2SAT è chiusa rispetto al complemento, è sufficiente esibire un gadget che riduca $f(a_1, a_2, a_3) = a_1 \wedge a_2 \wedge a_3$ a 2SAT. Tale gadget è

$$y, (\neg y \vee x_1), (\neg y \vee x_2), (\neg y \vee x_3),$$

dove tutte le clausole hanno peso 1. Le variabili x_1, x_2, x_3 sono variabili primarie e y è una variabile ausiliaria. □

Teorema 4.31 *Esiste un algoritmo polinomiale 2.7214-approssimato per* Max Weight 3Conj-SAT.

Corollario 4.32 *Esiste un algoritmo polinomiale 2.7214-approssimato per* Max Weight 3CSP.

Corollario 4.33 $\mathsf{PCP}_{c,s}[\log, 3] \subseteq \mathsf{P}$ *purché $c/s > 2.7214$.*

4.4 Limitazioni inferiori per costruzioni di gadget

In questa sezione mostreremo che alcune delle costruzioni di gadget menzionate in questa tesi e in [19] sono ottime, dimostreremo delle limitazioni inferiori per alcune altre costruzioni di gadget.

Il risultato seguente è utile per dimostrare limitazioni inferiori per la famiglia RMBC.

Lemma 4.34 *Se esiste un α-gadget che riduce un elemento della famiglia* RMBC *and una famiglia di vincoli $\mathcal{F}$ chiusa rispetto al complemento, allora esistono un α-gadget che riducono tutti gli elementi di* PC *a $\mathcal{F}$.*

Se there exists an α-gadget che riduce an element of the RMBC *family to a complementation-closed vincolo family $\mathcal{F}$, then there exists an α-gadget che riduce all elements of* PC *to $\mathcal{F}$.*

DIMOSTRAZIONE: Se una familgia $\mathcal{F}$ è chiusa rispetto al complemento, allora un α-gadget che riduce un elemento di PC (rispettivamente di RMBC) a $\mathcal{F}$ può essere modificato (tramite complementazione) e dare un un α-gadget che riduce tutti gli elementi di PC (rispettivamente di RMBC) a $\mathcal{F}$. Per questo motivo, restringeremo la nostra analisi ai soli gagdet per PC_0 e RMBC_{00}. Si noti che, per ogni $a_1, a_2, a_3 \in \{0,1\}^3$, $\mathrm{PC}_0(a_1, a_2, a_3) = 1$ se e solo se $\mathrm{RMBC}_{00}(a_1, a_2, a_3, \bar{a}_3) = 1$. Sia Γ un α-gadget su variabili promarie $x_1, \ldots, x_4$ e variabili ausiliarie $y_1, \ldots, y_n$ che riduce RMBC a 2SAT. Sia Γ' il gadget ottenuto da Γ imponendo $x_4 \equiv \bar{x}_3$: è immediato verificare che Γ' è un α-gadget che riduce PC_0 a $\mathcal{F}$. □

4.4.1 Ridurre PC e RMBC a 2SAT

Teorema 4.35 *Se Γ è an α-gadget che riduce un elemento* PC *di* 2SAT, *allora $\alpha \geq 11$.*

DIMOSTRAZIONE: È sufficiente considerare PC_0. Dimostriamo che l'ottimo di (PL1) è almeno 11. Per fare ciò, consideriamo il programma duale di (PL1). Abbiamo una variabile $y_{\vec{a},\vec{b}}$ per ogni $\vec{a} \in \{0,1\}^3$ e ogni $\vec{b} \in \{0,1\}^4$, più variabili aggiuntive $\hat{y}_{\vec{a},b(\vec{a})}$ per ogni $\vec{a} : \mathrm{PC}(\vec{a}) = 1$. La formulazione è

$$\begin{array}{lll}
\text{maximize} & \sum_{\vec{a},\vec{b}:\mathrm{PC}(\vec{a})=0} y_{\vec{a},\vec{b}} & \\
\text{subject to} & & \\
& \sum_{\vec{a},\vec{b}} y_{\vec{a},\vec{b}} \leq 1 + \sum_{\vec{a}:\mathrm{PC}(\vec{a})=1} y_{\vec{a},b(\vec{a})} & \\
& \sum_{\vec{a},\vec{b}} y_{\vec{a},\vec{b}} C_j(\vec{a},\vec{b}) \geq \sum_{\vec{a}:\mathrm{PC}(\vec{a})=1} \hat{y}_{\vec{a},b(\vec{a})} C_j(\vec{a}, b(\vec{a})) & \forall j \in [98] \\
& y_{\vec{a},\vec{b}} \geq 0 & (\forall \vec{a} \in \{0,1\}^3)\ (\forall \vec{b} \in \{0,1\}^4) \\
& \hat{y}_{\vec{a},b(\vec{a})} \geq 0 & (\forall \vec{a} : \mathrm{PC}(\vec{a}) = 1)
\end{array} \qquad \text{(DUAL1)}$$

Esiste una soluzione ammissibile di (DUAL1) il cui costo è 11, che può essere trovata risolvendo il programma lineare. □

Corollario 4.36 *Se Γ è un α-gadget che riduce un elemento di* RMBC *a* 2SAT, *allora $\alpha \geq 11$.*

4.4.2 Ridurre PC e RMBC a SAT

Teorema 4.37 *Se Γ è un α-gadget che riduce un elemento di* PC *a* SAT, *allora $\alpha \geq 4$.*

DIMOSTRAZIONE: Come nella prova del Teorema 4.35 è sufficiente osservare che una soluzione ammissibile per il duale fornisce una limitazione inferiore. Il programma lineare che trova il miglior gadget che riduce PC_0 a SAT è simile a (PL1), l'unica differenza è il maggior numero di clausole da considerare, cioè $N = \sum_{i=1}^{7} \binom{7}{i} 2^i$. Il programma duale è allora

$$\begin{array}{lll}
\text{maximize} & \sum_{\vec{a},\vec{b}:\mathrm{PC}(\vec{a})=0} y_{\vec{a},\vec{b}} & \\
\text{subject to} & & \\
& \sum_{\vec{a},\vec{b}} y_{\vec{a},\vec{b}} \leq 1 + \sum_{\vec{a}:\mathrm{PC}(\vec{a})=1} y_{\vec{a},b(\vec{a})} & \\
& \sum_{\vec{a},\vec{b}} y_{\vec{a},\vec{b}} C_j(\vec{a},\vec{b}) \geq \sum_{\vec{a}:\mathrm{PC}(\vec{a})=1} \hat{y}_{\vec{a},b(\vec{a})} C_j(\vec{a}, b(\vec{a})) & \forall j \in [N] \\
& y_{\vec{a},\vec{b}} \geq 0 & (\forall \vec{a} \in \{0,1\}^3)\ (\forall \vec{b} \in \{0,1\}^4) \\
& \hat{y}_{\vec{a},b(\vec{a})} \geq 0 & (\forall \vec{a} : \mathrm{PC}(\vec{a}) = 1)
\end{array} \qquad \text{(DUAL2)}$$

Si consideri il seguente assegnamento di valori alle variabili di (DUAL2) (i valori non specificati sono zero):

$$(\forall \vec{a} : \mathrm{PC}(\vec{a}) = 1)\ \hat{y}_{\vec{a},b(\vec{a})} = \tfrac{3}{4}$$
$$(\forall \vec{a} : \mathrm{PC}(\vec{a}) = 1)(\forall \vec{a}' : d(\vec{a},\vec{a}') = 1)\ y_{\vec{a}',b(\vec{a})} = \tfrac{1}{3}$$

dove d è la distanza di Hamming tra sequenze binarie. È possibile mostrare che si tratta di una soluzione ammissibile per ed è immediato verificare che il suo costo è 4. □

Corollario 4.38 *Se Γ è un α-gadget che riduce un elemento di* RMBC *a* SAT*, allora $\alpha \geq 4$.*

4.4.3 Ridurre PC e RMBC a CUT e DICUT

I seguenti teoremi possono essere dimostrati formulando il problema duale e trovandone una soluzione ottima.

Teorema 4.39 *Se Γ è un α-gadget che riduce un elemento di* PC *a* CUT*, allora $\alpha \geq 8$.*

Teorema 4.40 *Se Γ è un α-gadget che riduce un elemento di* RMBC *a* CUT*, allora $\alpha \geq 8$.*

Teorema 4.41 *Se Γ è un α-gadget che riduce un elemento di* PC *a* DICUT*, allora $\alpha \geq 6.5$.*

Teorema 4.42 *Se Γ è un α-gadget che riduce un elemento di* RMBC *a* DICUT*, allora $\alpha \geq 6$.*

4.4.4 Ridurre kSAT a lSAT

Siano k ed l due interi tali che $k > l \geq 3$. la riduzione standard da EkSAT a lSAT può essere vista come un $\lceil (k-2)/(l-2) \rceil$-gadget. In questa sezione mostreremo che questo è asintoticamente il meglio possibile. Si noti che, dato che lSAT è chiusa rispetto al complemento, possiamo restringerci a considerare solamente una funzione di vincolo di EkSAT, ad esempio $f(a_1,\ldots,a_k) \equiv \bigvee_i a_i$.

Teorema 4.43 *Per ogni $k > l > 2$, se Γ è un α-gadget che riduce f a lSAT allora $\alpha \geq k/l$.*

DIMOSTRAZIONE: Come mostrato nella Sezione 4.1.2, possiamo scrivere un programma lineare il cui ottimo dia il più piccolo α tale che esiste un α-gadget che riduce f a lSAT. Sia b la funzione testimone usata per formulare tale programma lineare. Possiamo assumere che b sia 2^{2^k}-aria, e fissiamo $K = 2^{2^k}$. Sia inoltre N il numero totale di vincoli da lSAT che possono essere definiti su $k + K$ variabili. Fissiamo una qualunque enumerazione $C_1,\ldots,C_N$ di tali vincoli. La formulazione duale è

$$\begin{array}{ll}
\text{maximize} & \sum_{\vec{a},\vec{b}:\mathrm{PC}(\vec{a})=0} y_{\vec{a},\vec{b}} \\
\text{subject to} & \\
 & \sum_{\vec{a},\vec{b}} y_{\vec{a},\vec{b}} \leq 1 + \sum_{\vec{a}:f(\vec{a})=1} y_{\vec{a},b(\vec{a})} \\
\forall j \in [N] & \sum_{\vec{a},\vec{b}} y_{\vec{a},\vec{b}} C_j(\vec{a},\vec{b}) \geq \sum_{\vec{a}:f(\vec{a})=1} \hat{y}_{\vec{a},b(\vec{a})} C_j(\vec{a},b(\vec{a})) \\
(\forall \vec{a} \in \{0,1\}^k)\ (\forall \vec{b} \in \{0,1\}^K) & y_{\vec{a},\vec{b}} \geq 0 \\
(\forall \vec{a} : f(\vec{a}) = 1) & \hat{y}_{\vec{a},b(\vec{a})} \geq 0
\end{array} \tag{DUAL3}$$

Isa $A_k \subset \{0,1\}^k$ l'insieme delle stringhe binarie di lunghezza k con esattamente una componente non nulla (chiaramente, $|A_k| = k$). Sia anche $\vec{0}$ (rispettivamente, $\vec{1}$) la stringa di lunghezza k tutte le cui componenti siano uguali a 0 (rispettivamente, 1). Quella che segue è una soluzione ammissibile per (DUAL3) il cui costo è k/l. Specifichiamo solamente i valori diversi da zero.

$$\begin{aligned}
&\forall \vec{a} \in A_k\ \hat{y}_{\vec{a},b(\vec{a})} = 1/l \\
&\forall \vec{a} \in A_k\ y_{\vec{0},b(\vec{a})} = 1/l \\
&\forall \vec{a} \in A_k\ y_{\vec{1},b(\vec{a})} = 1/k\ .
\end{aligned}$$

□

È interessante confrontare il risultato precedenre con il Lemma 3.24. In forza della limitazione inferiore precedente, una riduzione con gadget non può fornire una riduzione approximation-preserving da MAX SAT a MAX kSAT. Più in generale, non ci può essere una riduzione con gadget approximation-preserving da MAX SAT a MAX $(\log n)$SAT. Una tale riduzione può invece essere ottenuta con la nostra tecnica di "contrazione di disgiunzioni" e, in questo, abbiamo addirittura una riduzione *tight*, cioè che preserva la soglia di approssimazione.

Teorema 4.44 *Le soglie di approssimazione of* MAX SAT *e* MAX $k(n)$SAT *sono uguali, per funzione $k(n)$ che sia monotona non decrescente e illimitata superiormente.*

DIMOSTRAZIONE: È sufficiente dimostrare che per ogni r ed ogni $r' < r$, l'esistenza di un algoritmo r'-approssimato per MAX $k(n)$SAT implica l'esistenza di un algoritmo r-approssimato per MAX SAT.

Fissiamo p ed h tali che

$$r \geq \max\{1/(1-(1-p)^{h+1})\ ,\ r'/(1-p)\}\ .$$

Dal momento che $k(n)$ è non decrescente ed illimitata, esiste una certa costante n_0 (che dipende solo da h) tale che $k(n) \geq h$ per ogni $n \geq n_0$. L'algoritmo di approssimazione per MAX SAT usa la tecnica del Lemma 3.24 e un algoritmo r'-approssimato per MAX hSAT. Rimane da verificare che MAX hSAT sia r'-approximable. In effetti, le sole istanze di MAX hSAT che non siano anche istanze di MAX $k(n)$SAT hanno lunghezza al più $n_0 = O(1)$. Per tali istanze, una soluzione ottima può essere trovata in tempo costante. □

4.5 Eliminare i pesi dai problemi di soddisfacimento di vincoli

4.5.1 Pesi polinomialmente limitati e pesi arbitrari

Nel seguito considereremo problemi di ottimizzazione in cui l'obiettivo è trovare un insieme di oggetti che soddisfino una data proprietà e per i quali sia massima (o minima) la somma dei pesi. Più formalmente, diremo che un problema di ottimizzazione A è un *problema di sottoinsieme* se può essere espresso nel modo seguente.

- Ogni istanza x di A è una tripla (U, x', ω), dove $U = \{u_1, \ldots, u_n\}$ è un insieme di oggetti, x' è una stringa (eventualmente vuota), e $\omega : U \to N$ assegna un peso $\omega(u)$ ad ogni oggetto $u \in U$.

- Ad ogni soluzione $y \in \mathsf{sol}(x)$ è associato un sottoinsieme $S \subseteq U$, e la misura di y e la misura di y è uguale a $\sum_{u \in S} \omega(u)$.

Un problema di sottoinsieme A è *ben posto* se, per ogni istanza (U, x', ω), se y è una soluzione ammissibile, allora, per ogni funzione $\omega' : U \to N$, y è una soluzione ammissibile per (U, x', ω'). In altre parole, questa proprietà assicura che la definizione di soluzione ammissibile in A è indipendente dai pesi. È facile vedere che, ad esempio, Min Weight Vertex Cover e tutti i problemi Max Weight $\mathcal{F}$ sono problemi di sottoinsieme ben posti.

Sia A un problema di sottoinsieme ben posto e sia p un polinomio. Denotiamo con A^p la restrizione di A ad istanze x tali che la somma dei pesi sia al più $p(|x|)$. Si noti che per ogni famiglia di vincoli $\mathcal{F}$, Max $\mathcal{F} \equiv^1_{\mathrm{AP}}$ Max Weight $\mathcal{F}^{n^\epsilon}$ per ogni $\epsilon > 1$.

La dimostrazione del seguente risultato impiega una tecnica di scalatura simile a quella usata per dimostrare il Teorema 3.22.

Teorema 4.45 (Teorema di scalatura) *Sia A un problema di sottoinsieme ben posto in* APX *e sia $r > 1$. Esiste un polinomio p tale che se A^p è r-approssimabile, allora A è $(r + 1/n)$-approximable.*

Dimostrazione: Assumiamo che A sia un problema di massimizzazione (la dimostrazione per problemi di minimizzazione è simile). Sia A r_0-approssimabile, sia T un algoritmo polinomiale r_0-approssimato per A, e sia $t(x) = \mathsf{m}(x, T(x))$. Per ogni istanza x, abbiamo che

$$t(x) \leq \mathsf{opt}(x) \leq r_0 t(x).$$

Sia $x = (U, x', \omega)$ una istanza di taglia n (in particolare $|U| \leq n$). Definiamo una nuova funzione peso $\tilde{\omega}$, tale che, per ogni $u \in U$,

$$\tilde{\omega}(u) = \left\lfloor \frac{\omega(u) n^2}{t(x)} \right\rfloor.$$

Dal momento che A è ben posto, l'insieme delle soluzioni ammissibili per (U, x', ω) è uguale a quello di $(U, x', \tilde{\omega})$. Inoltre, possiamo assumere senza perdita di generalità che $\omega(u) \leq r_0 t(x)$ per ogni $u \in U$. Ne consegue che $\tilde{\omega}(u) \leq r_0 n^2$ per ogni $u \in U$. Perciò, $\tilde{x} = (U, x', \tilde{\omega})$ è una istanza di $A^{r_0 n^3}$.

Dalla definizione di $\tilde{\omega}$ segue che, per ogni soluzione ammissibile y,

$$\mathsf{m}(x, y) \geq \mathsf{m}(\tilde{x}, y) t(x)/n^2.$$

Inoltre,

$$\begin{aligned}
\mathsf{m}(x, y) &= \sum_{u \in S} \omega(u) \leq \sum_{u \in S} \frac{t(x)}{n^2} \left\lceil \frac{\omega(u) n^2}{t(x)} \right\rceil \\
&\leq (t(x)/n^2) \sum_{u \in S} \left(\left\lfloor \frac{\omega(u) n^2}{t(x)} \right\rfloor + 1 \right) \\
&= (t(x)/n^2)(\mathsf{m}(\tilde{x}, y) + |S|) \\
&\leq (t(x)/n^2)(\mathsf{m}(\tilde{x}, y) + n).
\end{aligned}$$

D'altra parte, possiamo assumere che $\mathsf{m}(x, y) \geq t(x)$, altrimenti sostituiamo y con la soluzione calcolata dall'algoritmo di approssimazione T. Allora,

$$
\begin{aligned}
\frac{\mathsf{opt}(x)}{\mathsf{m}(x,y)} &\le \frac{(t(x)/n^2)(\mathsf{opt}(\tilde{x})+n)}{\mathsf{m}(x,y)} \\
&\le \frac{(t(x)/n^2)\mathsf{opt}(\tilde{x})}{(t(x)/n^2)\mathsf{m}(\tilde{x},y)} + \frac{t(x)}{n\cdot\mathsf{m}(x,y)} \\
&\le \frac{\mathsf{opt}(\tilde{x})}{\mathsf{m}(\tilde{x},y)} + \frac{1}{n}.
\end{aligned}
$$

Perciò, se y è una soluzione r-approssimata per $\tilde{x}$, allora y è una soluzione $(r+\frac{1}{n})$-approssimata per x. Il teorema è dunque dimostrato. $\square$

Corollario 4.46 *Sia A un problema di sottoinsieme ben posto tale che $A \in \mathsf{APX}$. Allora esiste un polinomio p tla eche A e A^p hanno la stessa soglia di approssimazione.*

4.5.2 Il problema Max Cut

In questa sezione mostreremo come ridurre il problema di r-approssimare Max Weight Cut^p al problema di $(r-o(1))$-approssimare il problema Max Simple Cut, per ogni polinomio p.

Definizione 4.47 (Grafo "mixing") *Un grafo bipartito $G = (V_1, V_2, E)$ è detto (n, w, δ, ϵ)-mixing se $|V_1| = |V_2| = n$, e per ogni due sottoinsiemi $A \subseteq V_1$, $B \subseteq V_2$ con almeno δn nodi, vale la seguente disuguaglianza*

$$\left| Cut(A,B) - w\frac{|A||B|}{n^2} \right| \le \epsilon w \frac{|A||B|}{n^2}$$

dove $Cut(A,B)$ denota il numero di archi tra i nodi di A e nodi in B.

Osserviamo che un grafo (n, w, δ, ϵ)-mixing ha almeno $(1-\epsilon)w$ e archi e al più $(1+\epsilon)w$ archi.

È ben noto che i grafi expander hanno delle buone proprietà di mixing. In effetti, la prova del seguente teorema (rimandata alla Sezione 4.7) usa una nota costruzione di grafi expander.

Teorema 4.48 (Costruzione di grafi mixing) *Per ogni $\epsilon, \delta > 0$, esistono due costranti c ed n_0 tali che per ogni $n > n_0$ e per ogni $w \ge cn$, un grafo (n, w, ϵ, δ)-mixing può essere costruito in tempo polinomiale in n.*

Siamo adesso pronti per formulare e dimostrare il risultato principale di questa sezione.

Teorema 4.49 *Per ogni $\alpha > 1$,* Max Weight Cut $\le^{\alpha}_{\mathrm{AP}}$ Max Simple Cut.

Dimostrazione: Sia $(G = (V, E), \omega)$ una istanza di Max Weight Cut, sia $r > 1$ una costante fissata. Senza perdere in egeneralità (vedi Teorema 4.45) possiamo assumere che esista un polinomio q tale che $\omega(e) \le q(|V|)$ per ogni edge $e \in E$.

Fissiamo ϵ e δ tali che $0 < \epsilon \le 1/2$, $0 < \delta < 1/2$, e

$$\alpha \ge \frac{1}{r-1}\left(\frac{(1+\epsilon)}{(1-\epsilon)(1-12\delta)}r - 1\right) .$$

Siano c, n_0 le constanti del Teorema 4.48 relative ad ϵ e δ. Assumiamo che esista un intero $N > n_0$ tale che, per ogni arco e, $cN \le \omega(e) \le N^2$ (altrimenti possiamo moltiplicare ogni peso per $c^2 n_0^2 w_{\max}$, dove $w_{\max}$ è il peso massimo, e definire $N = cn_0 w_{\max}$).

Nel seguito $[N]$ indica l'insieme $\{1,\ldots,N\}$. Costruiamo un grafo non pesato $\hat{G} = (\hat{V}, \hat{E})$ nel modo seguente. Per ogni $cN \leq w \leq N^2$, sia $M_w = ([N],[N],E_w)$ il grafo (N, w, ϵ, δ)-mixing la cui esistenza è garantita dal Teorema 4.48. Definiamo $\hat{V}$ tale che contiene N copie di ogni vertice di V e, per ogni arco e di peso $\omega(e)$ di G, inseriamo in $\hat{E}$ gli archi di $M_{\omega(e)}$. Più formalmente,

$$\hat{V} = \{v^i \ : \ v \in V \text{ e } i = 1,\ldots,N\}$$

e

$$\hat{E} = \{(u^i, v^j) \ : \ (u,v) \in E \text{ e } (i,j) \in E_{\omega(u,v)}\}.$$

Fatto 4.50 *Per ogni taglio C in (G,ω) esiste un taglio $\hat{C}$ in $\hat{G}$ tale che*

$$\mathsf{m}(\hat{G},\hat{C}) \geq (1-\epsilon)\mathsf{m}(G,\omega,C)\ .$$

DIMOSTRAZIONE: [Del Fatto 4.50] Sia $\hat{C} = \{u^i \ : \ u \in C \text{ e } i \in [N]\}$. Allora,

$$\begin{aligned}
\mathsf{m}(\hat{G},\hat{C}) &= \sum_{(u,v)\in E, |\{u,v\}\cap C|=1} |E_{\omega(u,v)}| \\
&\geq \sum_{(u,v)\in E, |\{u,v\}\cap C|=1} (1-\epsilon)\omega(u,v) \\
&= (1-\epsilon)\mathsf{m}(G,\omega,C).
\end{aligned}$$

□

Fatto 4.51 *Supponiamo che, per ogni nodo u in G, siano dati due razionali $p_u, q_u \geq 0$ tali che $p_u + q_u = 1$, allora è possibile trovare in tempo polinomiale un taglio C tale che*

$$\mathsf{m}(G,\omega,C) \geq \sum_{(u,v)\in E} \omega(u,v)(p_u q_v + q_u p_v).$$

DIMOSTRAZIONE: [Del Fatto 4.51] Si consideri il taglio *probabilistico* nel quale ogni vertice u è scelto indipendentemente e con probabilità p_u. La misura media di tale tagli è chiaramente uguale a $\sum_{(u,v)\in E} \omega(u,v)(p_u q_v + q_u p_v)$. Usando il metodo delle probabilità condizionali (vedi ad esempio [4]), un taglio avente almeno tale misura può essere trovato in tempo polinomiale. □

Fatto 4.52 *Dato un taglio $\hat{C}$ in $\hat{G}$,possiamo costruire in tempo polinomiale un taglio C per G tale che $\mathsf{m}(G,\omega,C) \geq \mathsf{m}(\hat{G},\hat{C})(1-12\delta)/(1+\epsilon)$.*

DIMOSTRAZIONE: [Del Fatto 4.52] Assumiamo che $\hat{C}$ sia un ottimo locale, e perciò

$$\mathsf{m}(\hat{G},\hat{C}) \geq 1/2\sum_{e\in E}(1-\epsilon)\omega(e) \geq 1/4\sum_{e\in E}\omega(e).$$

Per prima cosa, trasformiamo $\hat{C}$ in un nuovo taglio $\hat{C}'$ tale che, per ogni $u \in V$, almeno δN copie di u appartengano a $\hat{C}'$ e almeno δN copie appartengano a $\hat{V} - \hat{C}'$. Chiaramente, $\mathsf{m}(\hat{G},\hat{C}')$ potrebbe essere inferiore a $\mathsf{m}(\hat{G},\hat{C})$. Comunque, le proprietà di mixing dei grafi $M_{\omega(u,v)}$ ci assicurano che

$$\begin{aligned}
\mathsf{m}(\hat{G},\hat{C}') &\geq \mathsf{m}(\hat{G},\hat{C}) - 2\sum_{e\in E}(1+\epsilon)\delta\omega(e) \\
&\geq \mathsf{m}(\hat{G},\hat{C})(1-12\delta).
\end{aligned}$$

Possiamo dunque costruire un taglio C tale che

$$\mathsf{m}(G,\omega,C) \geq \sum_{(u,v)\in E} \omega(u,v)(p_u(1-p_v)+p_v(1-p_u)).$$

D'altra parte, si consideri la misura di $\hat{C}'$: per ogni arco $(u,v)\in E$ dobbiamo considerare gli archi che connettono le Np_u copie di u in $\hat{C}'$ con le $N-Np_v$ copie di v non in $\hat{C}'$, più gli archi che connettono le $N-Np_u$ copie di u non in $\hat{C}'$ con le Np_v copie di v in $\hat{C}'$. Ricordando che tali archi sono presi da grafi mixing, abbiamo che

$$\begin{aligned}
\mathsf{m}(\hat{G},\hat{C}') &\leq \sum_{(u,v)\in E} \Big[\frac{(1+\epsilon)\omega(u,v)}{N^2}(Np_u(N-Np_v)) \\
&\quad + \frac{(1+\epsilon)\omega(u,v)}{N^2}(Np_v(N-Np_u))\Big] \\
&\leq (1+\epsilon)\mathsf{m}(G,\omega,C).
\end{aligned}$$

□

Dai fatti prcedenti, segue che data una soluzione $\hat{C}$ per $\hat{G}$ il cui rapporto di prestazione sia almeno r, possiamo costruire intempo polinomiale una soluzione C il cui rapporto di prestazione sia

$$\begin{aligned}
\frac{\mathsf{opt}(G,\omega)}{\mathsf{m}(G,\omega,C)} &\leq \frac{\mathsf{opt}(\hat{G})/(1-\epsilon)}{\mathsf{m}(\hat{G},\hat{C})(1-12\delta)/(1+\epsilon)} \\
&\leq \frac{1+\epsilon}{(1-12\delta)(1-\epsilon)} r \leq 1+\alpha(r-1).
\end{aligned}$$

Abbiamo dunque mostrato che MAX WEIGHT CUT $\leq_{\mathrm{AP}}^{\alpha}$ MAX SIMPLE CUT. □

Corollario 4.53 *Per ogni $\gamma > 0$, è* NP*-hard approssimare* MAX SIMPLE CUT *entro* $17/16-\gamma$.

4.5.3 Problemi di soddisfacimento di vincoli

Per trattare problemi MAX $\mathcal{F}$ dobbiamo introdurre la nozione di *insieme mixing* di tuple.

Definizione 4.54 (Insiemi mixing) *Siano $n>0$, $k>0$ interi positivi; un insieme $S\subseteq[n]^k$ è un insieme (n,w,ϵ,δ)-mixing k-ario se per ogni k sets $A_1,\ldots,A_k\subseteq[n]$, ognuno con almeno δN elementi, è valida la seguente disuguaglianza:*

$$\left| |S\cap A_1\times\cdots\times A_k| - w\frac{|A_1|\cdots|A_k|}{n^k}\right| \leq \epsilon w \frac{|A_1|\cdots|A_k|}{n^k}.$$

Per ogni k fissato e per valori sufficientememnte grandi di n e w, insiemi mixing k-ary sono efficientemente costruibili. Rimandiamo alla Sezione 4.7 la dimostrazione del seguente teorema.

Teorema 4.55 (Costruzione di insiemi mixing) *Presi comunque un intero $k\geq 2$ e due razionali $\epsilon,\delta>0$, esistono due costanti n_0 e c tali che, per ogni $n\geq n_0$ e per ogni w con $cn\leq w\leq n^k$, un insieme k-ario (n,w,ϵ,δ)-mixing è costruibile in tempo polinomiale in n.*

Teorema 4.56 *Sia $\mathcal{F}$ una famiglia di vincoli senza vincoli unari. Allora, per ogni $\alpha > 1$,* Max Weight $\mathcal{F} \leq^{\alpha}_{\mathrm{AP}}$ Max Simple $\mathcal{F}$.

Dimostrazione: La dimostrazione ha la stessa struttra della dimostrazione del Teorema 4.49. Ci restringiamo a Max Weight $\mathcal{F}^p$, per un opportuno polinomio p. Sia h l'arità più grande tra tutti i vincoli di $\mathcal{F}$. Siano ϵ, δ tali che

$$\alpha \geq \frac{1}{r-1}\left(\frac{(1+\epsilon)}{(1-\epsilon)(1+(1-(1-\delta)^h)2^h)}r - 1\right) .$$

Siano c, n_0 costanti tali che il Teorema 4.55 valga per ogni $k = 2, \ldots, h$. Sia $\phi = (C_1, \ldots, C_m, w_1, \ldots, w_m)$ una istanza di Max Weight $\mathcal{F}^p$ sull'insieme di variabili $X = \{x_1, \ldots, x_n\}$. Asumiamo che esista un intero $N > n_0$ tale che per ogni $i \in [m]$ valga $cN \leq w_i \leq N^2$ (altrimenti possiamo moltiplicare ogni peso per $c^2 n_0^2 w_{\max}$, dove $w_{\max}$ è il peso masimo, e poi definire $N = cn_0 w_{\max}$). Per ogni $cN \leq w \leq N^2$ e ogni $k \in [h]$, sia S_k^w un insieme k-ario (N, w, ϵ, δ)-mixing. Definiamo un'istanza $\hat{\phi}$ di Max Simple $\mathcal{F}$ nel modo seguente. L'insieme di variabili di $\hat{\phi}$ ha N copie di ogni variabile di X:

$$\hat{X} = \{x_i^j : i \in [n] \ , \ j \in [N]\} .$$

Per ogni vincolo k-ario $C_j = f(x_{i_1}, \ldots, x_{i_k})$ (con $f \in \mathcal{F}$) di peso w_j, definiamo un insieme di vincoli $\mathrm{red}(C)$ nel modo seguente:

$$\mathrm{red}(C) = \{f(x_{i_1}^{a_1}, \ldots, f(x_{i_k}^{a_k}) : (a_1, \ldots, a_k) \in S_k^{w_j}\} .$$

$\hat{\phi}$ è definito come l'unione degli insiemi $\mathrm{red}(C_j)$

$$\hat{\phi} = \bigcup_{j \in [m]} \mathrm{red}(C_j) .$$

Mostreremo nel seguito che ogni assegnamento per ϕ può essere trasformato in un assegnamento per $\hat{\phi}$ avente circa la stessa misura e viceversa.

Fatto 4.57 *Per ogni assegnamento τ per ϕ, esiste un assegnamento $\hat{\tau}$ per $\hat{\phi}$ tale che*

$$\mathsf{m}(\hat{\phi}, \hat{\tau}) \geq (1-\epsilon)\mathsf{m}(\phi, \tau) .$$

Dimostrazione: [Del Fatto 4.57] Definiamo $\hat{\tau}$ nel modo seguente: per ogni variabile $x_i^j \in \hat{X}$, $\hat{\tau}(x_i^j) = \tau(x_i)$. Abbiamo poi

$$\mathsf{m}(\hat{\phi}, \hat{\tau}) = \sum_j |\{C \in \mathrm{red}(C_j) : \hat{\tau} \text{ soddisfa } C\}| = \sum_{j : \tau \text{ soddisfa } C_j} |\mathrm{red}(C_j)| \geq (1-\epsilon)\mathsf{m}(\phi, \tau) .$$

□

Fatto 4.58 *Per ogni assegnamento $\hat{\tau}$ per $\hat{\phi}$, è possibile calcolare in tempo polinomiale un assegnamento τ per ϕ tale che*

$$\mathsf{m}(\phi, \tau) \geq \frac{1}{(1+\epsilon)(1+2^h(1-(1-\delta)^h))}\mathsf{m}(\hat{\phi}, \hat{\tau}) .$$

Dimostrazione: [Del Fatto 4.58] Sia $w_{\text{tot}} = \sum_j w_j$. Per cominciare, trasformiamo $\hat{\tau}$ in un assegnamento $\hat{\tau}'$ tale che per ogni variabile $x_i \in X$ almeno δN e al massimo $(1-\delta)N$ copie di x_i abbiamo valore true secondo $\hat{\tau}'$. Per fare questo, per ogni $x_i \in X$ che non soddisfa le condizioni richieste, cambiamo il valore di alcune delle sue copie. Si noti che, per ogni $x_i \in X$, viene cambiato il valore di al più δN delle sue copie. Questi cambiamenti possono essere tali che $\hat{\tau}'$ soddisfi meno vincoli di quanti non ne soddisfacesse $\hat{\tau}$. La differenza tra le due misure è limitata dal numero di vincoli dove occorre una variabile cambiata, e questo è al più $w_{\text{tot}}(1-(1-\delta)^h)(1+\epsilon)$:

$$\mathsf{m}(\hat{\tau}) - \mathsf{m}(\hat{\tau}') \leq w_{\text{tot}}(1-(1-\delta)^h)(1+\epsilon) \ .$$

Per ogni variable $x_i \in X$, sia

$$p_i = \frac{|\{j : \hat{\tau}'(x_i^j) = \mathsf{true}\}|}{N}$$

la frazione di copie di x_i il cui valore assegnato da $\hat{\tau}'$ è true. Definiamo τ_R come un *assegnamento probabilistico* su X nel modo seguente: $\mathbf{Pr}[\tau_R(x_i) = \mathsf{true}] = p_i$. Per ogni vincolo k-ario $C_j = f(x_{i_1}, \ldots, x_{i_k})$ di ϕ di peso w_j, sia $SAT_j \subseteq \{\mathsf{true}, \mathsf{false}\}^k$ l'insieme degli assegnamenti per le variabili di C_j che lo soddisfano. Dalla proprietà di mixing degli insiemi $S_k^{w_j}$ usati nella definizione di $\text{red}(C_j)$ è facile derivare

$$\begin{aligned}
\mathbf{Pr}(\tau_R \text{ soddisfa } C_j) &= \sum_{(b_1,\ldots,b_k)\in SAT_j} \mathbf{Pr}[\tau_R(x_{i_1} = b_1, \ldots, \tau_R(x_{i_k}) = b_k] \\
&= \sum_{(b_1,\ldots,b_k)\in SAT_j} \frac{|\{(a_1,\ldots,a_k) \in S_k^w : \hat{\tau}'(x_{i_1}^{a_1} = b_1, \ldots, x_{i_k}^{a_k} = b_k\}|}{N^k} \\
&\geq \frac{|\{C \in \text{red}(C) : \hat{\tau}' \text{ soddisfa } C\}|}{(1+\epsilon)w_j} \ .
\end{aligned}$$

E perciò abbiamo che

$$\mathsf{m}(\phi, \tau_R) \geq \mathsf{m}(\hat{\phi}, \hat{\tau}')/(1+\epsilon) \ .$$

Possiamo trovare in tempo polinomiale un assegnamento τ la cui misura è lameno la misura media di τ_R e almeno $2^{-h} w_{\text{tot}}$. Confrontiamo adesso la misura di τ con quella di $\hat{\tau}'$

$$\mathsf{m}(\phi, \tau) \geq \frac{\mathsf{m}(\hat{\phi}, \hat{\tau})}{1+\epsilon} - w_{\text{tot}}(1-(1-\delta)^h) \geq \frac{\mathsf{m}(\hat{\phi}, \hat{\tau})}{1+\epsilon} - 2^h(1-(1-\delta)^h)\mathsf{m}(\phi, \tau)$$

$$\mathsf{m}(\phi, \tau) \geq \frac{1}{(1+\epsilon)(1+2^h(1-(1-\delta)^h))} \mathsf{m}(\hat{\phi}, \hat{\tau})$$

□

Abbiamo dunque che, data una soluzione $\hat{\tau}$ per $\hat{\phi}$ il cui rapporto di prestazione sia al più r, possiamo costruire in tempo polinomiale una soluzione τ per ϕ il cui rapporto di prestazione è al più

$$\begin{aligned}\frac{\mathsf{opt}(\phi)}{\mathsf{m}(\phi,\tau)} &\le \frac{\mathsf{opt}(\phi)/(1-\epsilon)}{\mathsf{m}(\hat{\phi},\hat{\tau})/((1+\epsilon)(1+2^h(1-(1-\delta)^h)))} \\ &\le \frac{(1+\epsilon)(1+2^h(1-(1-\delta)^h))}{1-\epsilon} r \le 1+\alpha(r-1)\ .\end{aligned}$$

Abbiamo dunque mostrato che MAX WEIGHT $\mathcal{F} \le_{\text{AP}}^{\alpha}$ MAX SIMPLE $\mathcal{F}$. □

Corollario 4.59 *I seguenti problemi hanno la stessa soglia di approssimazione della loro versione pesata:* MAX SIMPLE DICUT*,* MAX SIMPLE EkSAT *per ogni* $k \ge 2$*,* MAX SIMPLE 2SAT*,* MAX SIMPLE k-CSP *per ogni* $k \ge 2$*.*

DIMOSTRAZIONE: L'unico risultato non banale è per MAX 2SAT (dato che 2SAT contiene vincoli unari). Usiamo un risultato di Yannakakis [110] che ha dimostrato che MAX WEIGHT 2SAT$\le_{\text{AP}}^{1}$ MAX WEIGHT E2SAT. Abbiamo dunque che, per ogni $\alpha > 1$,

$$\text{MAX WEIGHT 2SAT} \le_{\text{AP}}^{1} \text{MAX WEIGHT E2SAT} \le_{\text{AP}}^{\alpha} \text{MAX SIMPLE E2SAT}\ .$$

□

Il seguente corollario contiene una piccola raccolta dei risultati di non-approssimabilitá che seguono dal Teorema 4.56.

Corollario 4.60 *Per ogni* $\gamma > 0$*, è* NP*-hard approssimare* MAX SIMPLE DICUT *entro* $13/12-\gamma$*,* MAX SIMPLE E3SAT *(e perciò* MAX SIMPLE 3SAT *e* MAX SIMPLE SAT*) entro* $8/7-\epsilon$*,* MAX SIMPLE 2CSP *entro* $10/9-\gamma$*.*

4.6 Il problema Vertex Cover

4.6.1 Un altro problema di soddisfacimento di vincoli

I problemi MIN WEIGHT VERTEX COVER e MIN WEIGHT SAT sono entrambi ben posti. In questa sezione ci occuperemo principalemnte delle loro versioni non pesate e pesate con pesi polinomialmente limitati.

I due lemmi seguenti possono essere dimostrati con argomenti simili a quelli usati nella riduzione da MAX CLIQUE a PCP con zero free bit in [19] e nella riduzione da PCP con zero free bit a MAX CLIQUE data in [46], rispettivamente

Lemma 4.61 (Implicito in [19]) MIN VERTEX COVER$\le_{\text{AP}}^{1}$ MIN SAT.

Lemma 4.62 (Implicito in [46]) *Per ogni polinomio* p*,* MIN WEIGHT SAT^p $\le_{\text{AP}}^{1}$ MIN WEIGHT VERTEX COVER^p*.*

Il problema MIN VERTEX COVER può quindi essere visto come una riformulazione del problema MIN SAT.

4.6.2 Rimuovere i pesi

Lemma 4.63 *Per ogni polinomio p,* MIN WEIGHT VERTEX COVER$^p \leq^1_{\mathrm{AP}}$ MIN VERTEX COVER.

DIMOSTRAZIONE: Sia $G = (V, E)$ un grafo e $\omega : V \to N$ una funzione di peso sui suoi nodi, polinomialmente limitata. Definiamo il grafo $G^\omega = (V^\omega, E^\omega)$ nel modo seguente. Per ogni nodo $u \in V$ tale che $\omega(u) = w$, V^ω contiene w nodi distinti $u^1, \ldots, u^w$. Se $(u, v) \in E$ è un arco di G, allora E^ω contiene tutti gli archi (u^i, v^j) per $i = 1, \ldots, \omega(u)$ e $j = 1, \ldots, \omega(v)$. Notiamo che, dati G ed ω, G^ω può essere costruito in tempo polinomiale.

Sia C una soluzione per (G, ω) di misura c. Allora esiste una soluzione per G^ω di taglia c. Infatti, possiamo semplicemente considerare l'insieme $C^\omega = \{u^i \in V^\omega : u \in C \wedge i = 1, \ldots, \omega(u)\}$.

D'altra parte, sia C una soluzione per G^ω di taglia c. Da C, possiamo trovare in tempo polinomiale una soluzione C' per (G, ω) di misura almeno c. A questo scopo, assumiamo che un nodo $u \in V$ e due indici $i, j \in \{1, \ldots, \omega(u)\}$ esistano tali che $u^i \in C$ e $u^j \notin C$. Allora $C - \{u^i\}$ è ancora una soluzione ammissibile per G^ω. Infatti, u^i e u^j hanno lo stesso vicinato, e dato che $u^j \notin C$ ne consegue che tutti i nodi che sono adiacenti appartengono a C e perciò non è necessario prendere u^i nel ricoprimento. Un semplice argomento di induzione mostra che esiste un sottoinsieme C^ω di C che è ancora un ricoprimento tramite nodi e tale che per ogni $u \in U$ o tutte le sue copie appartengono a C^ω, o nessuna. C^ω può essere trasformato in una soluzione C' per (G, ω) di misura $|C^\omega| \leq c$. □

Dai tre lemmi precedenti otteniamo il seguente risultato.

Teorema 4.64 *Le soglie di approssimazione dei quattro problemi* MIN (WEIGHT) SAT *e* MIN (WEIGHT) VERTEX COVER *è la stessa.*

DIMOSTRAZIONE: Dal Corollario 4.46 segue l'esistenza di un polinomio p tale che MIN WEIGHT SAT^p e MIN SAT hanno la stessa soglia di approssimazione. I tre lemmi precedenti implicano che MIN WEIGHT SAT$^p \equiv^1_{\mathrm{AP}}$ MIN WEIGHT VERTEX COVER$^p \equiv^1_{\mathrm{AP}}$ MIN VERTEX COVER$\equiv^1_{\mathrm{AP}}$ MIN SAT. Questo conclude la dimostrazione del teorema. □

4.6.3 Ridurre il grado

In questa sezione mostreremo che un ingrediente importante nella prove del Teorema 2.22 è il fatto che un grafo FGLSS contiene molti sottografi bipartiti completi. Il nostro obiettivo è di rimpiazzare tali grafi bipartiti completi con grafi bipartiti sparsi che conservino una proprietà di connevità sufficientemente buona. Nel seguito definiamo questo tipo di grafi e mostriamo come generarli.

Definizione 4.65 (Switcher) *Sia $\epsilon > 0$ una costante positiva, siano n_1, n_2 interi positivi. Un grafo bipartito $G = (V_1, V_2, E)$ è un (n_1, n_2, ϵ)-switcher se le seguenti condiizoni sono verificate:*

1. $|V_1| = n_1$, $|V_2| = n_2$;

2. *per ogni ricoprimento tramite nodi C di G, o $|V_1 - C| \leq \epsilon|C|$ o $|V_2 - C| \leq \epsilon|C|$.*

In altre parole, uno switcher è tale che ogni ricoprimento tramite nodi deve scegliere quasi tutti i nodi di almeno una componente. Si osservi che un grafo bipartito completo con componenti di taglia n_1 e n_2 è un $(n_1, n_2, 0)$-switcher. Come sarà mostrato in seguito, i grafi bipartiti completi sono usati nella dimostrazione del Teorema 2.22 a causa delle loro perfette proprietà di switching.

Enunciamo adesso un risultato (la cui prova è riportate nella Sezione 4.7) sulla esistenza esulla costruzione esplicita di grafi con grado limitato e buone proprietà di switching.

Lemma 4.66 (Costruzione di switcher) *Esiste una costante $c > 0$ tale che, per ogni $\epsilon > 0$ e ogni n_1, n_2 tale $n_1 \geq n_2$, è possibile costruire in tempo polinomiale un (n_1, n_2, ϵ)-switchercon grado massimo $k \leq c(n_1 + n_2)/n_2\epsilon^2$.*

Teorema 4.67 (Non-approssimabilità di MIN VERTEX COVER-B **)**
Se $\mathsf{NP} \subseteq \mathsf{FPCP}_{c,s}[\log, f, q]$, alloro per ogni $\epsilon > 0$ esiste una costante $B = O(q^4/\epsilon^3)$ tale che è NP-hard approssimare MIN VERTEX COVER *in grafi di grado massimo B entro*

$$1 + \frac{c-s}{2^f - c} - \epsilon \ .$$

DIMOSTRAZIONE: Isa ϕ una istanza di SAT, e consideriamo il grafo FGLSS $G_\phi = (V_\phi, E_\phi)$. Tale grafo ha la seguente caratterizzazione. Sia l la lunghezza della prova a cui accede il verificatore; per ogni $i = 1, \ldots, l$, sia $\pi[i]$ l'i-esimo bit della prova π, e sia $U[i]$ (rispettivamente, $Z[i]$) l'insieme dei nodi del grafo corrispondenti a configurazioni in cui $\pi[i] = 1$ (rispettivamente, $\pi[i] = 0$). Sia inoltre $\mathbf{u}[i] = |U[i]|$ e $\mathbf{z}[i] = |Z[i]|$. Infine, sia $u_i^{(j)}$ (rispettivamente, $z_i^{(j)}$) il j-esimo elemento di $U[i]$ (rispettivamente, di $Z[i]$) in ordine lessicografico. Date due configurazioni (vertici) u e v, c'è un arco tra u e v se e solo se sono inconsistenti, cioè se e solo, per qualche i, leggono lo stesso bit $\pi[i]$ e si aspettano di ricevere valori differenti. Possiamo perciò caratterizzare l'insieme degli archi di G_ϕ come

$$E_\phi = \bigcup_{i=1}^{l} \{(u_i^{(j)}, z_i^{(k)}) : (j,k) \in K_{\mathbf{u}[i],\mathbf{z}[i]}\} \ ,$$

dove, per ogni n_1 e n_2, K_{n_1,n_2} è l'insieme degli archi del grafo bipartito completo con componenti $\{1, \ldots, n_1\}$ and $\{1, \ldots, n_2\}$. Inoltre, ogni nodo u di V_ϕ appartiene ad al più q insiemi $U[i], Z[i]$. Possiamo dunque vedere E_ϕ come l'unione di grafi bipartiti completi, cioè grafi con le migliori proprietà di switching possibili. Mostreremo ora che, nella medesima costruzione, è sufficiente usare switcher più deboli.

Senza perdere in generalità, assumiamo che per ogni $i = 1, \ldots, l$, $\mathbf{u}[i] \geq \mathbf{z}[i]$ (altrimenti possiamo invertire il valore dell'i-esimo bit della prova in ogni configurazione poi invertire i valori di $\mathbf{u}[i]$ e $\mathbf{z}[i]$). Sia γ una costante che sarà fissata in seguito, tale che $1/\gamma = O(q/\epsilon)$. Sia I l'insieme degli indici i tali che $\mathbf{z}[i] \geq \gamma(\mathbf{z}[i] + \mathbf{u}[i])$. Per ogni n_1 e per ogni n_2, sia S_{n_1,n_2} l'insieme degli archi di un (n_1, n_2, γ)-switcher (assumiamo che gli insiemi di vertici siano $\{1, \ldots, n_1\}$ e $\{1, \ldots, n_2\}$). Definiamo un grafo $G'_\phi = (V_\phi, E'_\phi)$ con lo stesso insieme di vertici di G_ϕ e con l'insieme di archi

$$E'_\phi = \bigcup_{i \in I} \{(u_i^{(j)}, z_i^{(k)}) : (j,k) \in S_{\mathbf{u}[i],\mathbf{z}[i]}\} \ .$$

Il Lemma 4.66 implica che, per ogni $i \in I$, possiamo costruire un $(\mathbf{u}[i], \mathbf{z}[i], \gamma)$-switcher con grado limitato da

$$O((\mathbf{u}[i] + \mathbf{z}[i])/\mathbf{z}[i]\gamma^2) = O(1/\gamma^3) \ .$$

Dato che ogni nodo appartiene ad al più q insiemi $U[i]$, $Z[i]$, e avevamo assunto che $\gamma = O(\epsilon/q)$, ne consegue che G'_ϕ ha grado limitato da $O(q^4/\epsilon^3)$.

Mostreremo ora come convertire ogni ricoprimento tramite nodi per G'_ϕ in un ricoprimento di poco più grande per G_ϕ. Per prima cosa, osserviamo che da ogni ricoprimento C' in G'_ϕ possiamo ottenere un ricoprimento C in G_ϕ tale che $|C| \leq |C'|(1 + q\gamma) + q\gamma n$. Infatti, sia $C'' = \bigcup_{i \notin I} Z[i]$: dalla definizione di I, segue che

$$|C''| \leq \sum_{i \notin I} \gamma(\mathbf{z}[i] + \mathbf{u}[i]) \leq q\gamma n \ .$$

Inoltre, per ogni $i \in I$, sia $C'[i] = C' \cap (U[i] \cup Z[i])$ l'insieme dei nodi di C' che sono usati per coprire le configurazioni dove viene letto l' i-esimo bit. Aggiungendo al più $\gamma C'[i]$ nodi, abbiamo un ricoprimento che comprende o tutti i nodi di $U[i]$ o tutti i nodi di $Z[i]$. Sia $C[i]$ questo nuovo insieme. Chiaramente, $C = C'' \cup \bigcup_{i=1}^{l} C[i]$ è un ricoprimento per G_ϕ, e abbiamo che

$$|C| - |C'| \leq q\gamma n + \sum_i |C[i]| - |C'[i]| \leq q\gamma n + q\gamma |C'|$$

dove $n = \mathbf{r}2^f$. Se ϕ non è soddisfacibile, allora

$$\mathsf{opt}(G'_\phi) \geq \frac{1}{1+q\gamma}\mathsf{opt}(G_\phi) - \gamma 2^f q\mathbf{r} \geq \mathbf{r}(2^f - s)\left(\frac{1}{1+q\gamma} - q\gamma\frac{2^f}{2^f - s}\right) \ .$$

Inoltre, G'_ϕ è un sottografo di G_ϕ con lo stesso insieme di vertici, perciè ogni ricoprimento di G_ϕ è anche un ricoprimento di G'_ϕ. Ne consegue che, se ϕ è soddisfacibile, allora

$$\mathsf{opt}(G'_\phi) \leq \mathsf{opt}(G_\phi) \leq \mathbf{r}(2^f - c) \ .$$

Prendendo γsufficientemente piccola (ma tale che $1/\gamma = O(q/\epsilon)$), otteniamo l'enunciato del teorema. □

Le nostre tecniche hanno anche implicazioni per il problema Min Vertex Cover in grafi con un numero non lineare di archi.

Corollario 4.68 (of Theorem 4.67) *Sia $h : \mathcal{Z}^+ \to \mathcal{Z}^+$ una funzione calcolabile tale che $\lim_n h(n) = \infty$, sia $\mathsf{NP} \subseteq \mathsf{FPCP}_{c,s}[\log, f, q]$. Allora, per ogni $\epsilon > 0$ il problema* Min Vertex Cover *ristretto a grafi di grado massimo $h(n)$ è NP-hard da approssimare entro*

$$1 + \frac{c-s}{2^f - c} - \epsilon \ .$$

Dimostrazione: Per ogni $\epsilon > 0$, il Teorema 4.67 implica che esiste una costante B_ϵ exists tale che il problema Min Vertex Cover è NP-hardda approssimare entro $1 - \epsilon + (c-s)/(2^f - c)$ anche quando ristretto a grafi di grado massimo B_ϵ. È chiaro che il problema Min Vertex Cover ristretto a grafi di grado massimo B_ϵ è riducibile al problema Min Vertex Cover ristretto a grafi di grado massimo $h(n)$. Infatti, i due problemi differiscono solo in un numero finito di istanze. □

4.7 Grafi mixing, insiemi mixing e switcher

4.7.1 Costruzione di grafi mixing

Per prima cosa, ricodiramo il seguente risultato ben noto sulle proprietà di mixing dei grafi espander.

Lemma 4.69 (Vedi ad esempio [58]) *Sia $G = (V, E)$ un grafo d-regolare, sia λ un limite superiore al valore assoluto di tutti gli autovalori, salvo il più grande, della matrice di adiacenza di G, siano A e B sottoinsiemi disgiunti di vertici, allora*

$$\left| Cut(A,B) - \frac{d}{2|V|}|A||B| \right| \leq \lambda\sqrt{|A||B|}.$$

L'autovalore più grande della matrice di adiacenza di un grafo d-regolare è d, perciò abbiamo bisogno di grafi con secondo autovalre considerevolmente più piccolo di d. I gradi di Ramanujan sono adatti ai nostri scopi.

Teorema 4.70 ([87], Teorema 4.4) *Siano p e q primi tali che $p \equiv q \equiv 1 \pmod 4$ e $q > p$. Allora esiste un grafo $Y^{p,q}$ che è $p+1$-regolare, ha $q+1$, e tale che il secondo autovalore più grande della sua matrice di adiacenza è al più $2\sqrt{p}$.*

La versione di Dirichlet del Teorema dei Numeri Primi ci aiuta a capire la densità dei numeri primi della forma $4k+1$. Sia $\pi_{b,c}(n)$ il numero di primi $p \leq n$ tali che $p \equiv c \pmod b$, e sia $\phi(n)$ la funzione di Eulero, allora vale il seguente teorema.

Teorema 4.71 (Prime Numbers Theorem) *Se b e c are co-prime, then*

$$\lim_{n\to\infty} \frac{\pi_{b,c}(n)}{n/\ln n} = \frac{1}{\phi(b)}.$$

Il teorema precedente è dovuto a Dirichlet, Hadamard, e de la Vallèe Poussin, e una dimostrazione può essere trovata in [41]. Sia ora $l_{b,c}(n)$ il più piccolo primo $p \geq n$ tale che $p \equiv c \pmod b$.È un facile esercizio mostrare che dal teorema precedente segue che

$$\lim_{n\to\infty} \frac{l_{4,1}(n)}{n} = 1.$$

In altre parole, per ogni ϵ esiste una costante n_0 tale che per ogni $n \geq n_0$ esiste un primo $n \leq p \leq n(1+\epsilon)$ tale che $p \equiv 1 \pmod 4$.

Possiamo adesso dimostrare il Teorema 4.48.

DIMOSTRAZIONE: [Del Teorema 4.48] Sia n_0 tale che per ogni $m \geq 2n_0 - 1$ esiste un primo $p \equiv 1 \pmod 4$ tale che $m \leq p \leq m(1+\epsilon/2)$. Sia $c = \max\{n_0, 32/(\epsilon^2\delta^2)\}$.

Si consideri un qualunque $n \geq n_0$ e w tale che $cn \leq w \leq n^2$. Se $w > n^2/(1+\epsilon/2)$ allora il grafo bipartito completo $\mathrm{K}_{n,n}$ è chiaramente (n, w, ϵ, δ)-mixing. Altrimenti, le condizioni precedenti implicano che esistono primi $p \equiv q \equiv 1 \pmod 4$ tali che

$$n - 1 \leq q \leq (n-1)(1+\epsilon/2) \leq n(1+\epsilon/2) - 1,$$

$$2w/n - 1 \leq p \leq (2w/n - 1)(1+\epsilon/2) \leq 2w/n(1+\epsilon/2) - 1.$$

Cioè,

$$\frac{2w}{n^2(1+\epsilon/2)} \leq \frac{p+1}{q+1} \leq \frac{2w(1+\epsilon/2)}{n^2}$$

e quindi

$$\left|\frac{p+1}{2(q+1)} - \frac{w}{n^2}\right| \leq \frac{\epsilon}{2}\frac{w}{n^2}. \tag{4.5}$$

Consideriamo adesso il grafo $Y^{p,q}$; siano V_1, V_2 due insiemi disgiunti di n vertici di $Y^{p,q}$. Sia E l'insieme degli archi di $Y^{p,q}$ che abbiamo un estremo in V_1 e un estremo in V_2, dimostreremo che il grafo bipartito $G = (V_1, V_2, E)$ è (n, w, ϵ, δ)-mixing. Siano $A \subseteq V_1$ e $B \subseteq V_2$ due sottoinsiemi di almeno δn nodi, sia $a = |A|$, $b = |B|$. Applicando il Lemma 4.69 abbiamo che

$$\left|Cut(A,B) - \frac{p+1}{2(q+1)}ab\right| \leq 2\sqrt{pab}$$

e, usando (4.5),

$$\begin{aligned}
\left|Cut(A,B) - \frac{w}{n^2}ab\right| &\leq \frac{\epsilon}{2}\frac{w}{n^2}ab + 2\sqrt{pab} \\
&\leq \frac{\epsilon}{2}\frac{w}{n^2}ab + 2\sqrt{\frac{2w}{n}ab} \\
&\leq \frac{w}{n^2}ab\left(\frac{\epsilon}{2} + 2\sqrt{\frac{2n^3}{wab}}\right) \\
&\leq \frac{w}{n^2}ab\left(\frac{\epsilon}{2} + 2\sqrt{\frac{2n^3}{cn\delta^2 n^2}}\right) \\
&\leq \frac{w}{n^2}ab\left(\frac{\epsilon}{2} + \frac{\epsilon}{2}\right).
\end{aligned}$$

□

4.7.2 Costruzione di insiemi mixing

DIMOSTRAZIONE:[Del Teorema 4.55] Cominciamo con il dimostrare il teorema assumendo che $k = 2^h$ sia una potenza di due. Procediamo per induzione su h. Per $h = 1$ il teorema è equivalente al Teorema 4.48. Assumiamo che il teorema valga per per $k = 2^h$ e consideriamo il caso $2k = 2^{h+1}$. Siano $\epsilon, \delta > 0$ costanti positive arbitrarie, siano $\bar{\epsilon}, \bar{\delta} > 0$ costanti positive tali che

$$\frac{(1+\bar{\epsilon})^3}{(1-\bar{\epsilon})^2} - 1 \leq \epsilon \ ,\ 1 - \frac{(1-\bar{\epsilon})^3}{(1+\bar{\epsilon})^2} \leq \epsilon \text{ e } \bar{\delta} \leq \frac{(1-\bar{\epsilon})}{1+\bar{\epsilon}}\delta^k.$$

L'ipotesi induttiva ci assicura l'esistenza di costanti c, n_0 tali che per ogni $n, m > n_0$ e ogni $w > cm$, esiste un insieme $(n, cn, \bar{\epsilon}, \delta)$-mixing 2^h-ario. Siano $n \geq n_0$ e $w \geq c^2(1+\bar{\epsilon})n$ be interi, sia S un insieme k-ario $(n, cn, \bar{\epsilon}, \delta)$-mixing con m elementi e sia B un insieme 2-ario $(m, w, \bar{\epsilon}, \bar{\delta})$-mixing set. Si noti che B è efficientemente costruibile dato che

$$w \geq c^2(1+\bar{\epsilon})n \geq cm.$$

Fissiamo un ordine arbitrario (ad esempio quello lessicografico) tra le tuple di S, che sia un biiezione tra S e [m[. Usando questa biiezione, possiamo vedere le coppie di B come $2k$-tuple su $[n]$. Seguendo questo punto di vista, mostreremo che B è un insieme 2k-ario (n, w, ϵ, δ)-mixing.

Sia $A_1, \ldots, A_{2k} \subseteq [n]$ una famiglia di $2k$ insiemi tali che, per ogni $i = 1, \ldots, 2k$ $|A_i| \geq \delta n$. Definiamo due insiemi T e U tali che

$$T = S \cap (A_1 \times \cdots \times A_k) \text{ e } U = S \cap (A_{k+1} \times \cdots \times A_{2k}).$$

Osserviamo per prima cosa che, dalle proprietà di mixing di S, segue che

$$(1-\bar{\epsilon})cn \leq m \leq (1+\bar{\epsilon})cn,$$

che

$$(1-\bar{\epsilon})cn\frac{|A_1|\cdots|A_k|}{n^k} \leq |T| \leq (1+\bar{\epsilon})cn\frac{|A_1|\cdots|A_k|}{n^k},$$

e che

$$(1-\bar{\epsilon})cn\frac{|A_{k+1}|\cdots|A_{2k}|}{n^k} \leq |U| \leq (1+\bar{\epsilon})cn\frac{|A_{k+1}|\cdots|A_{2k}|}{n^k}.$$

Osserviamo che $|T|, |U| \geq \bar{\delta}m$ e perciò, dalla proprietà di mixing di B, otteniamo

$$(1-\bar{\epsilon})w\frac{|T||U|}{m^2} \leq |B \cap (T \times U)| \leq (1+\bar{\epsilon})w\frac{|T||U|}{m^2}.$$

Mettendo insieme tutti i pezzi abbiamo che

$$\begin{aligned}\frac{(1-\bar{\epsilon})^3}{(1+\bar{\epsilon})^2}w\frac{|A_1|\cdots|A_{2k}|}{n^{2k}} &\leq |B \cap (A_1 \times \cdots \times A_{2k})| \\ &\leq \frac{(1+\bar{\epsilon})^3}{(1-\bar{\epsilon})^2}w\frac{|A_1|\cdots|A_{2k}|}{n^{2k}}.\end{aligned}$$

Per concludere la dimostrazione, notiamo che da un insieme k-ario (n, w, ϵ, δ)-mixing S (dove $w \leq n^{k-1}$) possiamo ottenere un insieme $(k-1)$-ario (n, w, ϵ, δ)-mixing S' nel modo seguente

$$S' = \{(a_1, \ldots, a_{k-1}) : \exists a_{k+1}.(a_1, \ldots, a_{k-1}, a_k) \in S\}.$$

□

4.7.3 Costruzione di switcher

Per costruire switcher, è utile riformulare la proprietà (2) nella Definizione 4.65 in modo differente. Sia I un qualunque insieme indipendente in G, siano $A = V_1 \cap I$ e $B = V_2 \cap I$. Allora la proprietà (2) è equivalente a chiedere che o

$$|A| \leq \epsilon(|V_1| + |V_2| - |A| - |B|) ,$$

or

$$|B| \leq \epsilon(|V_1| + |V_2| - |A| - |B|) .$$

Se consideriamo la versione contropositiva dell'ultimo enunciato, abbiamo che la proprietà (2) vale se e solo se per ogni sottoinsieme $A \subseteq V_1$ e per ogni sottoinsieme $B \subseteq V_2$ tale che

$$|A|, |B| > \epsilon(n_1 + n_2 - (|A| + |B|))$$

esiste almeno un arco in E che collega un nodo in A con un nodo in B. Quest'ultima è in effetti una proprietà di espansione: gli switcher sono in effetti una generalizzazione degli *OR dispersers*.

Definizione 4.72 (OR disperser [104]) *Un (n_1, n_2, ϵ)-disperser è un grafo bipartito $G = (V_1, V_2, E)$ tale che $|V_1| = n_1$, $|V_2| = n_2$, e per ogni sottoinsieme $A \subseteq V_1$, $B \subseteq V_2$ tale che $|A| \geq \epsilon n_1$ e $|B| \geq \epsilon n_2$, esiste almeno un arco in E con un estremo in A e un estremo in B.*

Proposizione 4.73 *Un $(n_1, n_2, \epsilon/(1+\epsilon))$-OR disperser è anche un (n_1, n_2, ϵ)-switcher.*

DIMOSTRAZIONE: Sia $G = (V_1, V_2, E)$ un $(n_1, n_2, \epsilon/(1+\epsilon))$-OR disperser, e siano $A \subseteq V_1$ e $B \subseteq V_2$ tali che

$$\begin{aligned}|A| &> \epsilon(n_1 + n_2 - (|A| + |B|)) \\ |B| &> \epsilon(n_1 + n_2 - (|A| + |B|))\end{aligned}$$

Dato che $|B| \leq n_2$, abbiaom che $|A| \geq \epsilon(n_1 - |A|)$, cioè,

$$|A| \geq \frac{\epsilon}{1+\epsilon} n_1 \ .$$

In modo analogo possiamo mostrare che

$$|B| \geq \frac{\epsilon}{1+\epsilon} n_2 \ .$$

Di conseguenza, G è un (n_1, n_2, ϵ)-switcher. □

DIMOSTRAZIONE: [Del Lemma 4.66] È sufficiente esisibre una costruzione di (n_1, n_2, γ)-OR disperser dove $\gamma = \epsilon/(1+\epsilon) \geq \epsilon/2$. Sia G un expander di Ramanujan d-regolare con n nodes, dove $n_1 + n_2 \leq n \leq 4(n_1 + n_2)$, e $512(n_1+n_2)/n_2\gamma^2 < d \leq 2048(n_1+n_2)/n_2\gamma^2$. Il secondo autovalore più grande λ di G è al più $2\sqrt{d-1}$. Dal Lemma 4.69, abbiamo che il numero di archi che connettono A e B è almeno

$$d\frac{|A||B|}{2n} - \lambda\sqrt{|A||B|} \ .$$

Consideriamo adesso due insiemi A, B tali che $|A| \geq \gamma n_1$ e $|B| \geq \gamma n_2$. Abbiamo che

$$\begin{aligned}d\frac{|A||B|}{2n} - \lambda\sqrt{|A||B|} &= \sqrt{|A||B|d}\left(\frac{\sqrt{|A||B|d}}{2n} - \frac{\lambda}{\sqrt{d}}\right) \\ &> \sqrt{|A||B|d}\left(\frac{\sqrt{\gamma n_1 \gamma 512(n_1+n_2)/\gamma^2}}{2n} - 2\right) \\ &\geq \sqrt{|A||B|d}\left(\frac{\sqrt{256(n_1+n_2)^2}}{2n} - 2\right) \\ &\geq 0 \ .\end{aligned}$$

Siano ora V_1 e V_2due insiemi disgiunti di nodi di G tali che $|V_1| = n_1$ e $|V_2| = n_2$. Sia G' il sottografo bipartito di G indotto dalle componenti V_1 e V_2: chiaramente, G' è un (n_1, n_2, γ)-OR disperser (perciò un (n_1, n_2, ϵ)-switcher) e il suo grado massimo è $O((n_1+n_2)/n_2\epsilon^2)$. □

Capitolo 5

Conclusioni e problemi aperti

Plan, *v.t. To bother about the best method of accomplishing an accidental result.*

(A. Bierce [24])

Questa tesi ha avuto come argomento la complessità di problemi di approssimazione combinatoria. Sono stati presentati risulati *strutturali*, risultati di *non-approssimabilità*, e risultati *algoritmici.* Il tema unificante è stata la ricerca di nuovi tipi di riduzioni approximation-preserving, che superassero le inerenti limitazioni di riduzioni precedenti. Per dimostrare risultati di completezza in classi di approssimazione naturali abbiamo definito un nuovo tipo di riduzione che *conosce il fattore di approssimazione che deve conservare.* Sfruttando la generalità della nostra nuova definizione di riducibilità, abbiamo dimostrato un risultato di completezza in $\mathsf{Max\ NP}$ senza fare uso di tecniche PCP. Per comprendere le proprietà dei problemi completi in classi di approssimazione e le limitazioni di precedenti riducibilità, abbiamo usato risultati di *query complexity*, prendendo in prestito risultati e tecniche di dimostrazione della teoria strutturale della complessità.

Abbiamo dimostrato che la *programmazione lineare* può essere usata per costruire al computer riduzioni per *sostituzione locale* dimostrabilmente ottimali. Abbiamo mostrato come estendere risultati di non-approssimabilità da problemi pesati a problemi non pesati usando in un modo nuovo grafi expander e il metodo probabilistico. Abbiamo dato una nuova formulazione logica del problema Min Vertex Cover e abbiamo dimostrato come usare grafi expander per dimostrare un forte risultato di non-approssimabilità per la sua restrizione a grafi con grado limitato.

Il nostro risultato di completezza senza PCP e alcune delle nostre riduzioni locali ottime implicano i migliori algoritmi di approssimazione noti per Max Planar GSAT-B, Max 3Sat e Max 3CSP.

Vi sono ancora moltissime questioni che attendono di essere risolte nel campo della teoria dell'approssimazione. Nel seguito ne alenchiamo alcune.

Risultati strutturali

Migliori caratterizzazioni di problemi completi. Le caratterizzazzione dei problemi polyAPX- e APX-completi dare nella Sezione 3.4 non sono così naturali come ci piacerebbe che fossero. In particolare, per i problemi polyAPX-completi sarebbe meglio avere come condizione la $\mathsf{NPF}^{\mathsf{NP}[1]}$-hardness di A_{n^ϵ} per un certo $\epsilon > 0$ e poi una proprietà combinatoria analoga alla self-improvability. Ancora meglio, si potrebbe avere solo l'hardness di A_2 e poi usare la self-improvability per incrementare il gap, così come si fa in pratica per Max Clique. Alcune difficoltà tecniche ci hanno impedito di fare questo, ma speriamo che possano essere superate. Inoltre, non sappiamo se esista una caratterizzazione dei problemi APX PB-completi rispetto alla E-riducibilità. Una tale caratterizzazione potrebbe chiarificare meglio la relazione tra la E-riducibilità e la AP-riducibilità.

Problemi logAPX-completi. Ancora non abbiamo una caratterizzazione dei problemi logAPX-completi. Inoltre non è del tutto chiaro come mai i problemi logAPX-completi non sono self-improvable. Presumibilmente, può essere trovata una dimostrazione senza PCP del fatto che Min Set Cover non è self-improvable

Il teorema PCP e l'approssimabilità di Max Clique. Il prodotto di grafi randomizzato mostra che se il teorema PCP è vero, anche se con verificatori che hanno errore two-sided, allora Max Clique è difficile da approssimare entro n^δ per un certo $\delta > 0$. Non è nota la connessione inversa. Bellare, Goldreich e Sudan hanno dimostrato che la difficoltà di approssimare Max Clique è "equivalente" all'esistenza di verificatori per NP con piccola complessità di free bit ammortizzati. La loro connessione inversa è ottimale rispetto ai fattori di non-approssimabilità. Ci piacerebbe avere una connessione inversa (anche più debole) rispetto al parametro del numero di bit letti. Equivalentemente, vorremmo una riduzione semplice e diretta che affermi che per ogni $r > 1$, un $\delta_r > 0$ esiste tale che approssimare Max Sat entro r si riduce ad approssimare Max Clique entro n^{δ_r}.

Risultati di non-approssimabilità

Risultati "definitivi". Håstad [63] ha dimostrato che è NP-hard approssimare Max 3Sat entro un fattore $8/7 - \epsilon$; il miglior algoritmo di approssimazione noto (vedi Sezione 4.3) ha un rapporto di prestazione $5/4$. Un algoritmo migliore (che chiuda il gap) potrebbe usare idee collegate alle riduzioni approximation-preserving di questa tesi. Per Min Vertex Cover, il miglior algoritmo noto ottiene un fattore 2, e il risultato di non-approssimabilità di Håstad è $7/6 - \epsilon$. Un risultato di non-approsimabilità che usi il Teorema 2.22 e un verificatore PCP con completezza perfetta non può essere migliore di $3/2 - \epsilon$ (ciò è dovuto al fatto che un verificatore con completezza perfetta deve usare almeno $\log 3$ free bit, vedi [19] e la discussione in [18]). L'opinione corrente è che la soglia esatta sia 2, e saranno necessarie idee radicalmente nuove per ottenere un tale risultato di non-approssimabilità.

Risultati di non-approssimabilità per problemi pratici. Non si conosce nessuno risultato esplicito di non-approssimabilità per Min TSP in spazi metrici e Min Steiner Tree. Le riduzioni con gadget non sembrano essere applicabili, in quanto tali problemi non hanno la chiara definizione logica in termini di soddisfacimento di vincoli. La APX-completezza di Min Steiner Tree è dimostrata in [23] tramite una riduzione da Min Vertex Cover in grafi con grado limitato (in effetti, sarebbe sufficiente partire da grafi sparsi). Per avere un risultato di non-approssimabilità ragionevole per Min Steiner Tree sarebbe necessario trovare un risultato di non-approssimabilità esplicito per, ad esempio, Min Vertex Cover-4 o Min Vertex Cover-10.

Applicazioni algoritmiche

Schemi di approssimazione. La nostra tecnica di riduzione descritta nella Sezione 3.3 implica un PTAS efficiente per le versioni planari di Max Sat e Max GSAT-B. Khanna e Motwani [79] hanno sviluppato schemi di approssimazione per una classe più ampia di problemi definiti logicamente: forse, una migliore riduzione potrebbe avere come conseguenza schemi di approssimazione per altri problemi.

Vincoli unari. Ci piacerebbe dimostrare che la soglia di approssimazione di Max Sat è uguale a quella di Max Weight Sat. Un ostacolo verso la dimostrazione di questo risultato è il fatto che le clausole di Sat possono essere arbitrariamente grandi, ma possiamo usare il Lemma 3.24 per portare la taglia della disgiunzioni ad una costante. La parte più difficile è eliminare i vincoli unari. Una riduzione da Max Weight Sat a Max Weight Sat senza vincoli unari (ma anche una da Max Weight 3SAT a Max Weight 3SAT senza vincoli unari) avrebbe rilevanza algoritmica dato che i vincoli unari sono un collo di bottoglia nelle analisi di approssimazione in [58] e nella Sezione 4.3.

Bibliography

[1] D. Adams. *The Hitchhiker's Guide to the Galaxy.* Harmony Books, 1980.

[2] D. Adams. *The Restaurant at the End of the Universe.* Pan Books, 1980.

[3] P. Alimonti. Non-oblivious local search for graphs and hypergraphs coloring problems. In *Proceedings of the 21th Workshop on Graph-Theoretic Concepts in Computer Science*, pages 167–180. LNCS 1017, Springer-Verlag, 1995.

[4] N. Alon and J. Spencer. *The Probabilistic Method.* Wiley Interscience, 1992.

[5] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and hardness of approximation problems. In *Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science*, pages 14–23, 1992.

[6] S. Arora and S. Safra. Probabilistic checking of proofs; a new characterization of NP. In *Proceedings of the 33rd IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1992.

[7] S. Arora and M. Sudan. Personal Communication, 1996.

[8] G. Ausiello, A. D'Atri, and M. Protasi. Structure preserving reductions among convex optimization problems. *Journal of Computer and System Sciences*, 21:136–153, 1980.

[9] L. Babai. Trading group theory for randomness. In *Proceedings of the 17th ACM Symposium on Theory of Computing*, pages 421–429, 1985. See also [12].

[10] L. Babai, L. Fortnow, L. Levin, and M. Szegedy. Checking computations in polylogarithmic time. In *Proceedings of the 23rd ACM Symposium on Theory of Computing*, pages 21–31, 1991.

[11] L. Babai, L. Fortnow, and C. Lund. Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity*, 1:3–40, 1991. Preliminary version in *Proc. of FOCS'90*.

[12] L. Babai and S. Moran. Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences*, 36:254–276, 1988.

[13] B. Baker. Approximation algorithms for NP-complete problems on planar graphs. *Journal of the ACM*, 41(1):153–180, 1994.

[14] J.L. Balcázar, J. Díaz, and J Gabarró. *Structural Complexity I.* Springer-Verlag, 1988.

[15] T. Beherendt, K.J. Compton, and E. Grädel. Optimization problems: expressibility, approximation properties and expected asymptotic growth of optimal solutions. In *Proceedings of the 6th Workshop of Computer Science and Logic*, pages 43–60. LNCS 702, Springer-Verlag, 1993.

[16] R. Beigel. Bounded queries to SAT and the boolean hierarchy. *Theoretical Computer Science*, 84:199–223, 1991.

[17] M. Bellare. Interactive proofs and approximation: Reductions from two provers in one round. In *Proceedings of the 2nd IEEE Israel Symposium on Theory of Computing and Systems*, 1993. Preliminary version in IBM Research Report RC17969 (1992).

[18] M. Bellare. Proof checking and approximation: Towards tight results. *Sigact News*, 27(1), 1996.

[19] M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCP's and non-approximability – towards tight results (3rd version). Technical Report TR95-24, Electronic Colloquium on Computational Complexity, 1995. Preliminary version in *Proc. of FOCS'95*.

[20] M Bellare, S. Goldwasser, C. Lund, and A. Russell. Efficient probabilistically checkable proofs and applications to approximation. In *Proceedings of the 25th ACM Symposium on Theory of Computing*, pages 294–304, 1993. See also the errata sheet in *Proc of STOC'94*.

[21] M. Bellare and M. Sudan. Improved non-approximability results. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 184–193, 1994.

[22] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. *Information and Computation*, 96:77–94, 1992. Preliminary version in *Proc. of STACS'89*.

[23] M. Bern and P. Plassmann. The Steiner tree problem with edge lengths 1 and 2. *Information Processing Letters*, 32:171–176, 1989.

[24] A. Bierce. *The Devil's Dictionary*. Neale Publishing, 1911.

[25] A. Blum, M. Li, J. Tromp, and M. Yannakakis. Linear approximation of shortest superstrings. *Journal of the ACM*, 41(4):630–647, 1994.

[26] D.P. Bovet and P. Crescenzi. *Introduction to the Theory of Complexity*. Prentice Hall, 1993.

[27] R. Chang. A machine model for NP-approximation problems and the revenge of the boolean hierarchy. *Bulletin of the EATCS*, 54, 1994.

[28] R. Chang. On the query complexity of clique size and maximum satisfiability. In *Proceedings of the 9th IEEE Conference on Structure in Complexity Theory*, pages 31–42, 1994.

[29] R. Chang, W. I. Gasarch, and C. Lund. On bounded queries and approximation. Technical Report TR CS-94-05, Department of Computer Science, University of Maryland Baltimore County, 1994. To appear on SIAM Journal of Computing.

[30] A.E.F. Clementi and L. Trevisan. Improved non-approximability results for vertex cover with density constraints. In *Proceedings of 2nd Computing and Combinatorics Conference*, pages 333–342. LNCS 1090, Springer-Verlag, 1996.

[31] A. Condon. The complexity of the max-word problem and the power of one-way interactive proof systems. *Computational Complexity*, 3:292–305, 1993. Preliminary version in *Proc. of STACS91*.

[32] S.A. Cook. The complexity of theorem proving procedures. In *Proceedings of the 3rd ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[33] P. Crescenzi and V. Kann. A compendium of NP optimization problems. Technical Report TR SI-95/02, Università di Roma "La Sapienza", Dipartimento di Scienze dell'Informazione, 1995. Updated on-line version is available at the URL `http://www.nada.kth.se/~viggo/problemlist/compendium.html`.

[34] P. Crescenzi, V. Kann, R. Silvestri, and L. Trevisan. Structure in approximation classes. In *Proceedings of the 1st Combinatorics and Computing Conference*, pages 539–548. LNCS 959, Springer-Verlag, 1995.

[35] P. Crescenzi and A. Panconesi. Completeness in approximation classes. *Information and Computation*, 93:241–262, 1991. Preliminary version in *Proc. of FCT'89*.

[36] P. Crescenzi and R. Silvestri. Relative complexity of evaluating the optimum cost and constructing the optimum for maximization problems. *Information Processing Letters*, 33:221–226, 1990.

[37] P. Crescenzi, R. Silvestri, and L. Trevisan. To weight or not to weight: Where is the question? In *Proceedings of the 4th IEEE Israel Symposium on Theory of Computing and Systems*, pages 68–77, 1996.

[38] P. Crescenzi and L. Trevisan. On approximation scheme preserving reducibility and its applications. In *Proceedings of 14th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 330–341. LNCS 880, Springer-Verlag, 1994.

[39] P. Crescenzi and L. Trevisan. MAX NP-completeness made easy. Manuscript, 1996.

[40] E. Dahlhaus, D.S. Johnson, C.H. Papadimitriou, P.D. Seymour, and M. Yannakakis. The complexity of multi-way cuts. *SIAM Journal on Computing*, 23:864–894, 1994.

[41] H. Davenport. *Multiplicative Number Theory*. Springer-Verlag, 1980. Second Edition.

[42] J. Edmonds and R.M. Karp. Theoretical improvements in algorithmic efficiency for network flow problem. *Journal of the ACM*, 19(2):248–264, 1972.

[43] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In *SIAM-AMS Proceedings*, pages 43–73, 1974.

[44] U. Feige. A threshold of $\ln n$ for approximating set cover. In *Proceedings of the 28th ACM Symposium on Theory of Computing*, 1996.

[45] U. Feige and M. Goemans. Approximating the value of two provers proof systems, with applications to MAX 2SAT and MAX DICUT. In *Proceedings of the 3rd IEEE Israel Symposium on Theory of Computing and Systems*, pages 182–189, 1995.

[46] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy. Interactive proofs and the hardness of approximating cliques. *Journal of the ACM*, 43(2):268–292, 1996. Preliminary version in *Proc. of FOCS91*.

[47] U. Feige and J. Kilian. Two prover protocols - low error at affordable rates. In *Proceedings of the 26th ACM Symposium on Theory of Computing*, pages 172–183, 1994.

[48] U. Feige and J. Kilian. Zero knowledge and the chromatic number. In *Proceedings of the 11th IEEE Conference on Computational Complexity*, 1996.

[49] U. Feige and L. Lovàsz. Two-prover one round proof systems: Their power and their problems. In *Proceedings of the 24th ACM Symposium on Theory of Computing*, pages 733–741, 1992.

[50] L.R. Ford and D.R. Fulkerson. *Flows in Networks.* Princeton University Press, 1962.

[51] L. Fortnow, J. Rompel, and M. Sipser. On the power of multi-prover interactive protocols. In *Proceedings of the 3th IEEE Conference on Structure in Complexity Theory*, pages 156–161, 1988.

[52] M. Fürer and B. Raghavachari. Approximating the minimum degree spanning tree to within one from the optimal degree. In *Proceedings of the 3rd ACM-SIAM Symposium on Discrete Algorithms*, pages 317–324, 1992.

[53] M.R. Garey and D.S. Johnson. The complexity of near-optimal graph coloring. *Journal of the ACM*, 23:43–49, 1976.

[54] M.R. Garey and D.S. Johnson. Strong NP-completeness results: motivation, examples, and implications. *Journal of the ACM*, 25:499–508, 1978.

[55] M.R. Garey and D.S. Johnson. *Computers and Intractability: a Guide to the Theory of NP-Completeness.* Freeman, 1979.

[56] F. Gavril. Manuscript cited in [55], 1974.

[57] M. Goemans and D. Williamson. New 3/4-approximation algorithms for the maximum satisfiability problem. *SIAM Journal on Discrete Mathematics*, 7(4):656–666, 1994. Preliminary version in *Proc. of IPCO'93*.

[58] M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995. Preliminary version in *Proc. of STOC'94*.

[59] S. Goldwasser, S. Micali, and C. Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on Computing*, 18(1):186–208, 1989. Preliminary version in *Proc of STOC'85*.

[60] R.L. Graham. Bounds for certain multiprocessing anomalies. *Bell System Technology Journal*, 45:1563–1581, 1966.

[61] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. In *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science*, 1996.

[62] J. Håstad. Testing of the long code and hardness for clique. In *Proceedings of the 28th ACM Symposium on Theory of Computing*, 1996.

[63] J. Håstad. Some optimal inapproximability results. In *Proceedings of the 29th ACM Symposium on Theory of Computing*, 1997.

[64] L.A. Hemaspaandra. Complexity theory column 5: the not-ready-for-prime-time conjectures. *SIGACT News*, 1994.

[65] D.S. Hochbaum. A framework for half integrality and 2-approximations with applications to feasible cut and minimum satisfiability. Manuscript, 1996.

[66] D.S. Hochbaum and D.B. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985.

[67] I. Holyer. The NP-completeness of edge-coloring. *SIAM Journal on Computing*, 10:718–720, 1981.

[68] H. B. Hunt III, M.V. Marathe, V. Radhakrishnan, S.S. Ravi, D.J. Rosenkrantz, and R.E. Stearns. Approximation schemes using L-reductions. In *Proceedings of 14th Conference on Foundations of Software Technology and Theoretical Computer Science*, pages 342–353. LNCS 880, Springer-Verlag, 1994.

[69] O. H. Ibarra and C. E. Kim. Fast approximation for the knapsack and sum of subset problems. *Journal of the ACM*, 22:463–468, 1975.

[70] R. Impagliazzo and M. Naor. Decision trees and downward closures. In *Proceedings of the 3th IEEE Conference on Structure in Complexity Theory*, pages 29–38, 1988.

[71] D.S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.

[72] J. Kadin. The polynomial time hierarchy collapses if the boolean hierarchy collapses. *SIAM Journal of Computing*, 17(6):1263–1282, 1988.

[73] J. Kadin. Erratum: The polynomial time hierarchy collapses if the boolean hierarchy collapses. *SIAM Journal on Computing*, 20(2):404, 1990.

[74] V. Kann. Polynomially bounded minimization problems which are hard to approximate. *Nordic Journal of Computing*, 1:317–331, 1994. Preliminary version in *Proc. of ICALP'93*.

[75] V. Kann and A. Panconesi. *Hardness of Approximation*. Annotated bibliographies in combinatorial optimization, 1996. To appear.

[76] N. Karmakar and R.M. Karp. An efficient approximation scheme for the one-dimensional bin packing problem. In *Proceedings of the 23th IEEE Symposium on Foundations of Computer Science*, pages 312–320, 1982.

[77] R.M. Karp. Reducibility among combinatorial problems. In R.E. Miller and J.W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[78] S. Khanna. Personal Communication, 1996.

[79] S. Khanna and R. Motwani. Towards a syntactic characterization of PTAS. In *Proceedings of the 28th ACM Symposium on Theory of Computing*, 1996.

[80] S. Khanna, R. Motwani, M. Sudan, and U. Vazirani. On syntactic versus computational views of approximability. In *Proceedings of the 35th IEEE Symposium on Foundations of Computer Science*, pages 819–830, 1994.

[81] P.G. Kolaitis and M.N. Thakur. Logical definability of NP optimization problems. *Information and Computation*, 115(2):321–353, 1994.

[82] M.W. Krentel. The complexity of optimization problems. *Journal of Computer and System Sciences*, 36:490–509, 1988.

[83] J.B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the AMS*, 7:48–50, 1956.

[84] D. Lapidot and A. Shamir. Fully parallelized multi prover protocols for NEXPTIME. In *Proceedings of the 32nd IEEE Symposium on Foundations of Computer Science*, pages 13–18, 1991.

[85] L. A. Levin. Universal search problems. *Problemi Peredachi Informatsii*, 9:265–266, 1973.

[86] T.J. Long. On γ-reducibility versus polynomial time many-one reducibility. *Theoretical Computer Science*, 14:91–101, 1981.

[87] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8:261–277, 1988.

[88] C. Lund, L. Fortnow, H. Karloff, and N. Nisan. Algebraic methods for interactive proof systems. *Journal of the ACM*, 39(4):859–868, 1992. Preliminary version in *Proc of FOCS'90.*

[89] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *Journal of the ACM*, 41:960–981, 1994. Preliminary version in *Proc. of STOC'93.*

[90] M.V. Marathe and S.S. Ravi. On approximation algorithms for the minimum satisfiability problem. *Information Processing Letters*, 58(1):23–29, 1996.

[91] T. Ono, T. Hirata, and T. Asano. Approximation algorithms for the maximum satisfiability problem. In *Proceedings of Scandinavian Workshop on Algorithmic Theory.* LNCS, Springer-Verlag, 1996.

[92] P. Orponen and H. Mannila. On approximation preserving reductions: complete problems and robust measures. Technical Report C-1987-28, Department of Computer Science, University of Helsinki, 1987.

[93] A. Panconesi and D. Ranjan. Quantifiers and approximations. *Theoretical Computer Science*, 107:145–163, 1993. Preliminary version in *Proc. of STOC'90.*

[94] C. H. Papadimitriou and M. Yannakakis. Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440, 1991. Preliminary version in *Proc. of STOC'88.*

[95] C.H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[96] C.H. Papadimitriou and M. Yannakakis. On limited nondeterminism and the complexity of the VC dimension. In *Proceedings of the 8th IEEE Conference on Structure in Complexity Theory*, pages 305–312, 1993.

[97] C.H. Papadimitriou and M. Yannakakis. The travelling salesman problem with distances one and two. *Mathematics of Operations Research*, 18:1–11, 1993.

[98] A. Paz and S. Moran. Non deterministic polynomial optimization problems and their approximation. *Theoretical Computer Science*, 15:251–277, 1981.

[99] M. Queyranne. Bounds for assembly line balancing heuristics. *Operations Research*, 33:1353–1359, 1985.

[100] R. Raz and S. Safra. A sub-constant error-probability PCP characterization of NP (part i). Manuscript, October 1996.

[101] R. Raz and S. Safra. A sub-constant error-probability PCP characterization of NP (part ii). Manuscript, September 1996.

[102] S. Sahni and T. Gonzalez. P-complete approximation problems. *Journal of the ACM*, 23:555–565, 1976.

[103] A. Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992. Preliminary version in *Proc of FOCS'90.*

[104] M. Sipser. Expanders, randomness or time vs. space. In *Proceedings of the 1st IEEE Conference on Structure in Complexity Theory.* Springer Verlag, 1986.

[105] L. Trevisan. A note on minimum-area upward drawing of complete and fibonacci trees. *Information Processing Letters*, 57(5):231–236, 1996.

[106] L. Trevisan. Positive linear programming, parallel approximation, and PCP's. In *Proceedings of the 4th European Symposium on Algorithms*, pages 62–75. LNCS 1136, Springer-Verlag, 1996.

[107] L. Trevisan, G.B. Sorkin, M. Sudan, and D.P. Williamson. Gadgets, approximation, and linear programming. In *Proceedings of the 37th IEEE Symposium on Foundations of Computer Science*, pages 617–626, 1996.

[108] K. Wagner. Bounded query computations. In *Proceedings of 3rd Structure in Complexity Theory Conference*, pages 260–277, 1988.

[109] T.S. Wee and J.M. Magazine. Assembly line balancing as generalized bin packing. *Operations Research Letters*, 1:56–58, 1982.

[110] M. Yannakakis. On the approximation of maximum satisfiability. *Journal of Algorithms*, 17:475–502, 1994. Preliminary version in *Proc. of SODA'92.*

[111] D. Zuckerman. NP-complete problems have a version that is hard to approximate. In *Proceedings of the 8th IEEE Conference on Structure in Complexity Theory*, pages 305–312, 1993.