Tesina Algoritmi Paralleli e Distribuiti
Uno sguardo alle classi NC e RNC

Marcello Cerulo
m.cerulo1@libero.it
Matr. 11120069

Flavio Chierichetti
flavio@lightless.org
Matr. 693411

17 giugno 2005

1 Complessità Computazionale

Questa tesina presenta le classi computazionali NC e RNC, le "equivalenti" di P e RP nella computazione parallela.

La classe P contiene tutti i problemi decisionali (o, equivalentemente, linguaggi) risolvibili in tempo polinomiale da un normale algoritmo monoprocessore o, più formalmente, da una macchina di Turing.

La classe NP contiene tutti i problemi decisionali risolvibili in tempo polinomiale da un algoritmo monoprocessore non deterministico. Un algoritmo non deterministico accetta un certo input se in almeno una delle sue computazioni raggiunge uno stato di accettazione; rifiuta l'input altrimenti.

La classe RP contiente tutti i problemi decisionali risolvibili da un algoritmo random monoprocessore.

Un algoritmo random è un algoritmo che può compiere scelte random di probabilità $\frac{1}{2}$.

Un certo input viene rifiutato da un algoritmo random quando l'algoritmo lo rifiuta indipendentemente dai risultati delle scelte random; un input viene accettato se l'algoritmo, su quell'input, termina in uno stato di accettazione con probabilità maggiore od uguale ad $\frac{1}{2}$.

Ovviamente vale $P \subseteq RP$ (ogni algoritmo in P è completamente deterministico: se accetta un certo input, lo fa con probabilità 1).

Vale inoltre $RP \subseteq NP$: un qualunque algoritmo random può essere trasformato in un algoritmo non deterministico sostituendo le scelte casuali con "sdoppiamenti" non deterministici della computazione.
Invece di lanciare una moneta virtuale e decidere di conseguenza come proseguire, si prosegue in tutti i modi possibili.
È chiaro che un input è accettato dall'algoritmo non deterministico se e solo se tale input è accettato da quello random.

Nel calcolo parallelo P ha un ruolo simile a quello di NP nel calcolo seriale. I problemi P-completi, sono i problemi più difficili di P: quelli che non si credono risolvibili in tempo polilogaritmico con un algoritmo parallelo.

Analogamente, esistono "somiglianze" tra NC (i problemi risolvibili velocemente nel parallelo) e P (quelli risolvibili velocemente nel seriale), o tra RNC

e RP: in particolare valgono le relazioni NC $\subseteq$ RNC $\subseteq$ RP come le relative nel seriale.

Nei prossimi paragrafi si illustreranno le classi NC e RNC.

2 La classe NC

In questo paragrafo si introdurrà la classe NC, presentandone alcuni aspetti fondamentali.

Definizione 2.1. *Sia $L \subseteq \Sigma^*$ un linguaggio sull'alfabeto Σ. L è in NC se esiste un algoritmo parallelo che $\forall w \in \Sigma^*$ decide, in tempo polilogaritmico in $|w|$ con un numero di processori polinomiale in $|w|$, se w è in L.*

2.1 Funzioni booleane

Una funzione booleana è:

- una variabile booleana;
- un'espressione della forma $\neg\phi$ dove ϕ è un'espressione booleana;
- un'espressione della forma $\phi_1 \wedge \phi_2$ dove ϕ_1 e ϕ_2 sono espressioni booleane;
- un'espressione della forma $\phi_1 \vee \phi_2$ dove ϕ_1 e ϕ_2 sono espressioni booleane.

Un circuito booleano è un grafo aciclico $C = (V, E)$ tale che

- tutti i nodi del grafo hanno grado entrante uguale a 0, 1 o 2;
- ogni nodo $i \in V$ ha associato un attributo $S(i) \in \{\text{true}, \text{false}, \wedge, \vee, \neg\} \cup \{x_1, x_2, ...\}$ tale che se $S(i) \in \{\text{true}, \text{false}\} \cup \{x_1, x_2, ...\}$ allora il nodo i non ha archi entranti;
- se $S(i) = \neg$ il nodo i ha un solo arco entrante;
- se $S(i) = \{\wedge, \vee\}$ il nodo i ha due archi entranti;
- esiste un unico nodo senza archi uscenti.

I nodi di un circuito booleano sono detti gate.

Un gate i senza archi entranti è chiamato input di C. Il gate senza archi uscenti è chiamato output di C.

La figura 1 mostra il circuito corrispondente all'espressione: $(x_1 \wedge x_2) \vee (x_2 \vee x_3)$.

Il valore di verità $T(i)$ di un gate $i \in V$ è definito induttivamente nel modo seguente:

- se $S(i) = \text{true}$ allora $T(i) = \text{true}$ ed equivalentemente per il false;
- se $S(i) \in \{x_1, x_2, ...\}$ allora $T(i) = x_i$;
- se $S(i) = \neg$ allora esiste un unico gate j tale che $(j, i) \in E$ e $T(i) = \neg T(j)$;
- se $S(i) = \wedge$ allora esistono due soli gate j e h tali che $(j, i), (h, i) \in E$ e $T(i) = T(j) \wedge T(h)$;

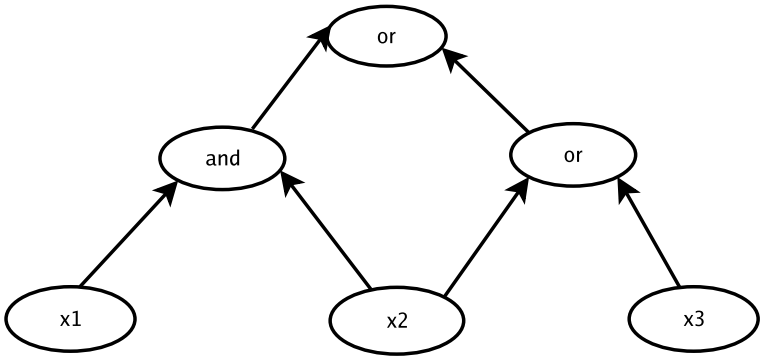

Figura 1: Circuito booleano.

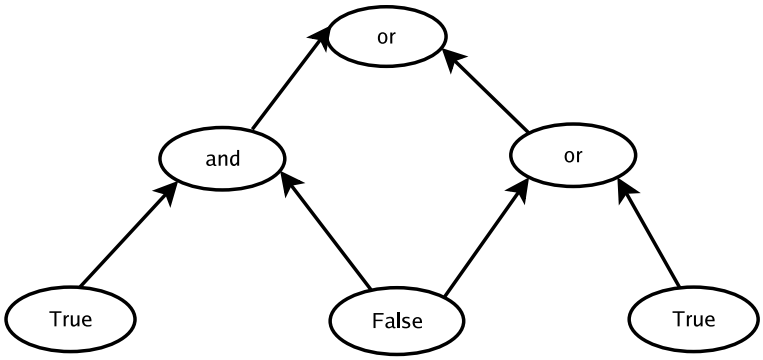

Figura 2: Circuito privo di variabili.

- Se $S(i) = \vee$ allora ci sono due gate j e h tali che $(j,i)e(h,i) \in E$ e $T(i) = T(j) \vee T(h)$.

Il valore del circuito $T(C)$ è uguale a $T(n)$ dove n è il gate di output.

Un circuito privo di variabili è un circuito tale che per ogni $i \in V$, $S(i) \notin \{x_1, x_2, ...\}$.

In fig. 2 è mostrato un circuito privo di variabili.

Teorema 2.1. *Il problema di determinare il valore $T(C)$ di un circuito C privo di variabili è P-completo.*

Dimostrazione. Ci si limiterà a provare che il problema è in P, per l'apparteneza a P-hard si veda [P94].

Poiché il grafo è orientato ed aciclico, è possibile ordinare topologicamente i nodi (in modo che per ogni arco (i,j) si abbia $i < j$).

Per eseguire l'ordinamento topologico basta visitare il grafo in profondità.

Una volta numerati i nodi, basta calcolare il valore $S(i)$ di ogni gate in ordine crescente di i. □

2.2 Funzioni booleane monotone

Una funzione booleana monotona è una funzione booleana con la proprietà che se uno degli input cambia da false a true, il valore della funzione non può cambiare da true a false.

Lemma 2.2. *f è una funzione monotona booleana se e solo se può essere espressa mediante un circuito booleano con soli* and *e* or*.*

Dimostrazione. Se un circuito è composto da soli and e or allora scambiando da false a true una qualsiasi variabile di ingresso, nessuno, tra i gate che l'hanno come input, potrà passare da valore true a valore false (nel caso di gate and o il valore rimarrà uguale o passerà da false a true, nel caso di gate or il valore diverrà certamente true).

Per induzione, tale proprietà vale su tutti i gate raggiungibili dalla variabile il cui valore è per ipotesi modificato: quindi se un circuito è composto unicamente da and e or esso descrive una funzione monotona booleana.

Viceversa, se f è una funzione monotona booleana nelle variabili $x_1, ..., x_n$ deve valere la proprietà che cambiando il valore di una variabile falsa, f non può variare da true a false.

Siano $(\hat{x}_1^1, ..., \hat{x}_n^1), ..., (\hat{x}_1^m, ..., \hat{x}_n^m)$ gli unici assegnamenti alle variabili $x_1, ..., x_n$ che rendono f vera. Allora f può essere espressa come

$$f' = \bigvee_{i=1}^{m} \left(\bigwedge_{\hat{x}_j^i = \text{true}} x_j \right)$$

Tale espressione è banalmente corretta: per costruzione, ogni tupla che rende f vera rende anche f' vera. Inoltre, per la monotonicità, le tuple che fanno assumere valore true ad f' hanno lo stesso effetto su f. □

Teorema 2.3. *Il problema del valore di un circuito di una funzione monotona è P-completo.*

Dimostrazione. Il problema del valore di un circuito booleano è P-completo. Si vuole dimostrare che ogni circuito booleano può essere ricondotto ad un circuito booleano con soli and e or. Dato un circuito booleano si possono portare indietro i $\neg$ verso i gate di input applicando le leggi di De Morgan:

$$\neg(a \vee b) = \neg a \wedge \neg b$$

$$\neg(a \wedge b) = \neg a \vee \neg b$$

□

2.3 Reti di Flusso e Taglio massimo

Una rete $N = (V, E, s, t, c)$ è un grafo orientato (V, E) con due nodi speciali sorgente e pozzo $s, t \in V$.

Una capacità c è una funzione $c : V \times V \to \mathbf{R}$ tale che $c(i, j) \geq 0$.

Un flusso in N è una funzione $f : V \times V \to \mathbf{R}$ che soddisfa le seguenti proprietà:

- $\forall (i, j) \in E \; f(i, j) \leq c(i, j)$;

- $\forall (i,j) \in E\ f(i,j) = -f(j,i)$;
- $\forall i \in V - \{s,t\}$ si richiede che $\sum_{(i,j)\in E} f(i,j) = 0$ (proprietà di conservazione del flusso);

Il valore del flusso f è definito come $|f| = \sum_{(s,i)\in E} f(s,i)$.

Dati $X, Y \subseteq V$ sia $F(X,Y) = \sum_{i\in X, j\in Y} f(i,j)$ dove $f(i,j) = 0$ se $(i,j) \notin E$.

Un taglio (S,T) di una rete di flusso $G = (V,E,s,t,c)$ è una partizione di V in S e $T = V - S$ tale che $s \in S$ e $t \in T$.

$F(S,T)$ sarà il flusso netto attraverso il taglio (S,T).

Lemma 2.4. *Sia f un flusso in una rete G con sorgente s e pozzo t e sia (S,T) un taglio di G, allora $F(S,T) = |f|$.*

Dimostrazione. Essendo $V = S \cup T$,

$$F(S,T) = \sum_{i\in S, j\in T} f(i,j) = \sum_{i\in S, j\in V} f(i,j) - \sum_{i,j\in S} f(i,j) = F(S,V) - F(S,S)$$

$F(S,S)$ è necessariamente nullo poiché nella sommatoria, per ogni $f(i,j)$, è presente anche $f(j,i) = -f(i,j)$.

Inoltre, per la la proprietà di conservazione del flusso, $F(S - \{s\}, V) = 0$, quindi

$$F(S,V) = F(\{s\},V) + F(S - \{s\},V) = F(\{s\},V) = |f|$$

ed il lemma è provato. □

Corollario 2.5. *Il valore di un qualunque flusso f in una rete di flusso G è limitato superiormente dalla capacità di un qualunque taglio di G.*

Dimostrazione. Sia (S,T) un qualunque taglio di G e sia f un qualunque flusso. Per il lemma precedente si ha

$$|f| = F(S,T) = \sum_{u\in S v\in T} f(u,v) \leq \sum_{u\in S v\in T} c(u,v) = c(S,T)$$

È ovvio che se $|f| = c(S,T)$ allora f è massimo. □

Il problema che ci si pone è quello di valutare se il flusso massimo di una rete N è dispari.

Teorema 2.6. *Il problema della disparità del flusso massimo è P-completo.*

Dimostrazione. Un'algoritmo polinomiale per la valutazione del flusso massimo è l'algoritmo di Ford-Fulkerson. Una volta ottenuto il flusso massimo, è possibile in tempo costante valutarne la disparità.

L'algoritmo di Ford-Fulkerson è il seguente:

- si inizializzi il flusso a 0
- finché esiste un cammino aumentabile (composto da soli archi non saturi) da s a t
  - si aumenti di a il flusso di ogni arco del cammino, dove a è la capacità residua minima tra gli archi di tale cammino

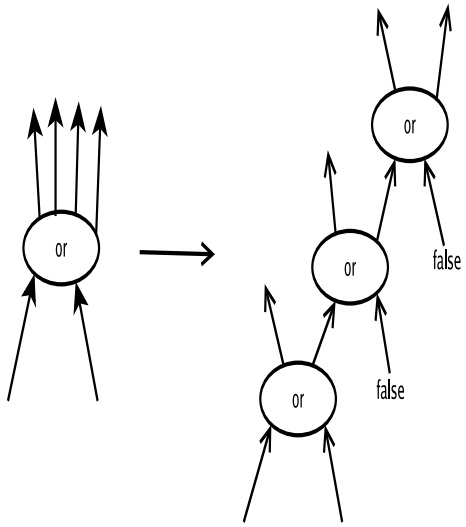

Figura 3: Trasformazione

- si restituisca il flusso ottenuto

Ad ogni passo l'algoritmo satura almeno un arco, quindi l'algoritmo termina in tempo polinomiale.

Lemma 2.7. *Se una rete G non contiene cammini aumentabili allora il flusso è massimo.*

Dimostrazione. Si consideri il taglio (S, T) della rete G, con:

$$S = \{v \in V - \{t\} \mid \text{esiste un cammino aumentabile da } s \text{ a } v \text{ in } G\} \cup \{s\}$$

$$T = V - S$$

$\forall (v, v') \in E$, con $v \in S - \{s\}, v' \in T - \{t\}$, si ha $f(v, v') = c(v, v')$: se così non fosse, esistendo un cammino aumentabile da s a v, esisterebbe un cammino aumentabile da s a v', il quale dovrebbe far parte di S. Contraddizione.

Per il corollario 2.5 si ha la tesi. □

Si ridurrà il problema della valutazione del circuito booleano monotono al problema della disparità del flusso massimo per dimostrare che quest'ultimo è P-arduo.

Sia C un circuito booleano monotono. Si assuma che il gate di output di C sia un or e che i gate di C non abbiamo più di due archi uscenti.

Tale assunzione non fa perdere perdere in generalità poiché si può trasformare un gate con più archi in uscita in un gate con solo due archi in uscita come illustrato in figura 3.

Si assuma che i gate di C siano numerati in modo tale che ogni gate abbia un indice minore dei predecessori.

Si aggiunga all'insieme V dei nodi della nostra rete un nodo speciale s (sorgente) e un nodo speciale t (pozzo). L'insieme V sarà quindi composto dai gate di C, con la loro numerazione, dal nodo sorgente e dal nodo pozzo.

L'insieme E degli archi della rete è composto invece da tutti gli archi del circuito C più un arco che collega la sorgente ad ogni gate true.

La capacità degli archi che collegano la sorgente ai gate true è uguale a $d2^i$ dove i è il numero corrispondente al gate true e d è il numero degli archi in uscita del gate true. La capacità degli archi che collegano un g_i ad un g_j è uguale a 2^i. Con g_i si intende il gate corrispondente al numero i. Sia g_h un generico gate and o or. Siano g_i e g_j i suoi predecessori. Il gate g_h avrà due archi in entrata con capacità 2^j e 2^i e al più due archi in uscita con capacità 2^h. Poiché i e j sono i numeri corrispondeti ai predecessori di g_h, $2^i + 2^j \geq 2^h + 2^h$.

Si definisca $s(g_h) = 2^i + 2^j - 2 \cdot 2^h$ con $s(g_h) \geq 0$. Se a g_h corrisponde un or verrà creato un arco (g_h, s) con capacità $s(g_h)$. Se a g_h corrisponde un and verrà creato un arco (g_h, s) con capacità $s(g_h)$. Si aggiunga un arco di capacità 1 per collegare il gate di output a t.

Un gate g è chiamato pieno rispetto ad un flusso f, se $\forall(g, k) \in E$ con $k \neq s, t$ si ha $f(g, k) = c(g, k)$.

Un gate g è chiamato vuoto rispetto ad un flusso f, se $\forall(g, k) \in E$ con $k \neq s, t$ si ha $f(g, k) = 0$.

Un flusso f è chiamato standard se rende tutti i gate true pieni e tutti i gate false vuoti.

Si dimostrerà ora che un flusso standard esiste e che cercarlo corrisponde a cercare un flusso massimo. Si costruisca un flusso standard cominciando dalla sorgente:

- sia e un arco in uscita dalla sorgente; il valore ad esso assegnato sarà pari alla capacità di e.
- sia e un arco in uscita da un gate true; il valore ad esso assegnato sarà pari alla capacità di e.
- sia e un arco in uscita da un gate false; il valore ad esso assegnato sarà pari a zero.
- sia g_h un gate or che ha valore uguale a true e i cui predecessori sono o gate di tipo true o gate di tipo false. Siano g_i e g_j i predecessori di g_h. Si avranno, allora, due casi:
  - uno dei due archi in entrata in g_h ha valore non nullo;
  - entrambi gli archi hanno valore diverso da zero.

  Per il primo caso, sia (g_i, g_h) l'arco in entrata con valore diverso da zero. Per costruzione sappiamo che il valore dell'arco (g_i,g_h) è 2^i (cioè pari alla sua capacità). Per come abbiamo numerato i gate $i \geq h + 1$, quindi si ha che $2^i \geq 2^h \cdot 2$; ma 2^{h+1} è uguale al flusso massimo in uscita da g_h poiché g_h può avere al più due archi in uscita, ciascuno di capacità 2^h. Si ha dunque:
  - $2^i - 2^{h+1} = w$ con $w \geq 0$ (nel caso che g_h abbia due archi in uscita);
  - $2^i - 2^h = w$ con $w > 0$ (nel caso che g_h abbia un solo arco in uscita).

  Siano $f(g_h, s) = w$ e $f(g_h, g_k) = 2^h \ \forall(g_h, g_k) \in E$.

  Per il secondo caso (entrambi gli archi hanno valore diverso da zero), si ha:

- $2^i + 2^j - 2^{h+1} = w$ con $w \geq 0$ (nel caso che g_h abbia due archi in uscita);
- $2^i + 2^j - 2^h = w$ con w>0 (nel caso che g_h abbia un solo arco in uscita).

Allora $f(g_h, s) = w$ e $f(g_h, g_k) = 2^h \; \forall (g_h, g_k) \in E$

- Sia g_h un gate or con valore uguale a false e i cui predecessori sono gate di tipo false. g_h non ha flusso entrante quindi assegniamo valore zero ai suoi archi in uscita.

- sia g_h un gate and con valore uguale a true e i cui predecessori sono gate di tipo true. Siano g_i e g_j i predecessori di g_h. Il flusso in entrata in g_h è uguale a $2^i + 2^j$. Come detto precedentemente $2^i + 2^j - 2^h = w$ con $w > 0$. Siano $f(g_h, t) = w$ e $f(g_h, g_k) = 2^h \; \forall (g_h, g_k) \in E$.

- sia g_h un gate and con valore uguale a false e i cui predecessori sono gate di tipo true o false. Siano g_i e g_j i predecessori di g_h. Poiché g_h ha valore uguale a false, uno solo degli archi in entrata in g_h avrà valore diverso da zero. Senza perdita di generalità si assuma sia g_i. Sia $f(g_h, t) = 2^i$.

Tutti i gate con adiacenti gate true o false sono stati considerati. Sostituendo al generico gate g il corrispettivo valore (true o false) si può ripetere lo stesso procedimento fino ad ottenere un flusso standard f. Si dimostrerà ora che il flusso standard ha valore massimo. Si supponga di dividere i nodi in due insiemi S, T:

- S contiene s e tutti i gate che hanno valore true
- T contiene t e tutti i gate che hanno valore false

(S, T) è un taglio della rete; il valore del flusso nella rete è uguale a $F(S, T)$. Sia (i, j) un generico arco tale che i e j non appartengano entrambi a S o a T e che $f(i, j) > 0$. L'arco (i, j) può solo essere:

- o un arco che collega un nodo or con valore vero ad un nodo and con valore falso;
- o un arco che collega un nodo true ad un nodo and con valore falso;
- o un arco che collega un nodo and con valore vero a t;
- o un arco che collega il nodo di output, se esso ha valore vero, con t.

Per costruzione, quindi, $f(i, j) = c(i, j)$; ciò implica che $F(S, T) = C(S, T)$. Per il corollario 2.5 f è massimo. Per come abbiamo costruito il flusso, $f(i, j)$, con $j \neq t$, sarà sempre pari. Quindi il flusso sarà dispari se e solo se il nodo di output avrà valore vero. □

In figura 4 è possibile vedere il grafo corrispondente al circuito in figura 2. L'etichetta dell'arco è uguale al rapporto tra il valore dell'arco e la capacità.

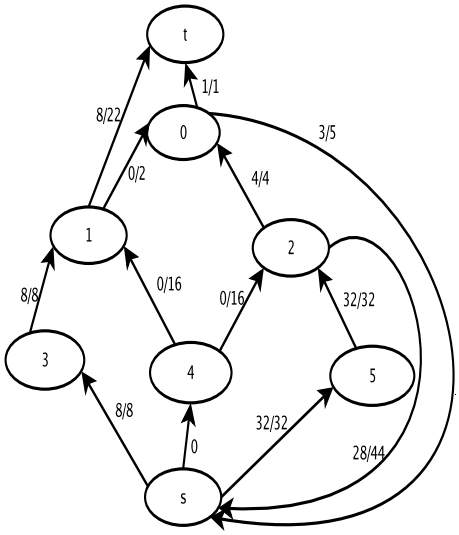

Figura 4: flusso

3 La classe RNC

In questo paragrafo si considererà la forza espressiva che è possibile aggiungere a NC utilizzando algoritmi paralleli random.

Un algoritmo parallelo random è un algoritmo parallelo che può eseguire un numero polinomiale di scelte random.

Definizione 3.1. *Sia $L \subseteq \Sigma^*$ un linguaggio sull'alfabeto Σ. L è in RNC se esiste un algoritmo parallelo random che $\forall w \in \Sigma^*$ decide, in tempo polilogaritmico in $|w|$ con un numero di processori polinomiale in $|w|$, se w è in L.*

3.1 Matching perfetto

Sia dato un grafo bipartito $G = (V \cup W, E)$ con nodi $V = \{v_1, ..., v_n\}$ e $W = \{w_1, ..., w_n\}$ ed archi $E \subseteq \{\{v, w\} \mid v \in V \wedge w \in W\}$.

Un matching perfetto in G è una permutazione π di $\{1, ..., n\}$ tale che $\{v_i, w_{\pi(i)}\} \in E$. Intuitivamente si vuole accoppiare i nodi in V, W evitando sovrapposizioni.

Il problema del matching perfetto è un caso particolare del problema del flusso massimo; per ottenere un'istanza del flusso massimo, si orientino gli archi di E in modo che vadano dall'insieme V all'insieme W, si aggiungano due nodi sorgente e pozzo s, t insieme agli archi (s, v) $\forall v \in V$ e (w, t) $\forall w \in W$; infine si pongano le capacità degli archi tutte uguali ad 1.

È chiaro che nel nuovo grafo esiste un flusso di valore n se e solo se in G esiste un matching perfetto.

3.2 Esistenza di un matching perfetto in G

In questo paragrafo si mostreranno i passi concettuali utili per la costruzione di un algoritmo parallelo random per la valutazione dell'esistenza di un matching

perfetto in un grafo G. Si proverà quindi che il problema del matching perfetto è in RNC.

3.2.1 Matrice simbolica M^G

Si consideri la matrice quadrata M^G di lato n che, in posizione i, j, abbia la variabile $t_{i,j}$ se $v_i, w_j \in E$ o 0 altrimenti.

Il determinante di M^G sarà

$$\det M^G = \sum_{\pi \in S_n} \sigma(\pi) \prod_{i=1}^{n} M^G_{i,\pi(i)}$$

dove S_n è il gruppo simmetrico di ordine n e $\sigma(\pi)$ è il segno della permutazione π (1 se la permutazione è pari, 0 se è dispari).

Ogni monomio nell'espressione del determinante è determinato dalla permutazione che l'ha generato. Inoltre esiste un certo monomio nell'espressione del determinante se nessuna posizione della matrice coinvolta da quel monomio contiene uno 0; o, in altre parole, se la permutazione che determina quel monomio è un matching perfetto.

Risulta quindi banale che $\det M^G$ è identicamente 0 se e solo se non esistono matching perfetti nel grafo G.

3.2.2 Matrice numerica N^G

Il prossimo passo trasformerà la matrice simbolica M^G nella matrice numerica N^G, mantenendo con alta probabilità la proprietà che $\det N^G = 0$ se e solo se G ha un matching perfetto.

Lemma 3.1. *Sia $P(t_1, ..., t_n)$ un polinomio (non costante) nelle variabili $t_1, ..., t_n$ ognuna con grado massimo d e sia $l > 0$ un intero. Il numero di tuple $(\hat{t}_1, ..., \hat{t}_n) \in \{0, 1, ..., l-1\}^n$ tali che $P(\hat{t}_1, ..., \hat{t}_n) = 0$ è al più ndl^{n-1}.*

Dimostrazione. Se $n = 1$ il teorema si riduce al teorema fondamentale dell'algebra: un polinomio di grado d non può avere più di d radici distinte.

Per $n \geq 2$, consideriamo P come un polinomio nella sola variabile t_n con le altre variabili usate come coefficienti delle varie potenze di t_n. Si supponga che tale polinomio abbia valore 0 in qualche punto di coordinate in $\{0, 1, ..., l-1\}^n$. In tali coordinate il coefficiente della potenza di grado più alto di t_n può avere o non avere valore 0. Consideriamo il caso in cui sia 0.
Quel coefficiente è un polinomio sulle variabili $t_1, ..., t_{n-1}$ con grado massimo d: per induzione, quindi, può avere al più $(n-1)dl^{n-2}$ zeri. Considerando che i possibili valori di t_n sono l, il polinomio su $t_1, ..., t_n$ può avere al più $(n-1)dl^{n-1}$ zeri.

Se invece il valore di quel coefficiente non è 0, si ha un polinomio di grado minore o uguale a d che può avere al più d radici per ogni possibile valore da associare a $t_1, ..., t_{n-1}$, quindi dl^{n-1}.

Sommando si ottiene il limite superiore. □

È possibile ora dare un algoritmo che decida il linguaggio (od il problema) dei grafi bipartiti senza matching perfetti.

- Si scelgano in modo casuale n interi tra 0 e $2n-1$.

- Si ottenga N^G da M^G sostituendo, per ogni i, t_i con l'i-esimo intero scelto.
- Se $\det N^G = 0$ allora G non ha un matching perfetto con probabilità almeno $\frac{1}{2}$.
- Se $\det N^G \neq 0$ allora G ha un matching perfetto.

Si noti che - per come è definito RNC - è necessario chiedersi se non esistano matching perfetti, piuttosto che la domanda diretta e naturale.

Un limite superiore per la probabilità p di ottenere un falso "positivo" si ottiene dividendo il limite superiore degli assegnamenti che annullano il determinante (se esso non è identicamente nullo) per il numero totale degli assegnamenti:

$$p \leq \frac{n(2n)^{n-1}}{(2n)^n} = \frac{n^n 2^{n-1}}{n^n 2^n} = \frac{1}{2}$$

La probabilità di ottenere un vero "positivo" è quindi limitata inferiormente da $\frac{1}{2}$.

È evidente che reiterando l'algoritmo k volte, vale la seguente proprietà:

- se almeno una volta $\det N^G$ risulta diverso da 0 allora G ha un matching perfetto;
- se $\det N^G = 0$ in ogni esecuzione, allora con probabilità almeno $1 - \frac{1}{2^k}$ G non ha un matching perfetto.

3.2.3 Determinante di matrici numeriche

Il determinate di matrici numeriche può essere calcolato in tempo $O(\log^2 n)$ con un numero polinomiale di processori: è quindi in NC.

È evidente che NC $\subseteq$ RNC: un algoritmo in NC accetta o rifiuta una parola indipendentemente da scelte casuali (che infatti non sono mai eseguite).
L'algoritmo in NC che segue prova quindi l'appartenenza a RNC del problema del matching perfetto.

Se A è una matrice quadrata di ordine n, con $A[i]$ si indica la sottomatrice quadrata di lato i, ottenuta eliminando le prime $n - i$ righe e colonne da A.

Secondo la regola di Cramer:

$$(A[i]^{-1})_{1,1} = \frac{\det A[i-1]}{\det A[i]}$$

Essendo $A[n] = A$, si ha che:

$$\det A = \left(\prod_{i=1}^{n} (A[i]^{-1})_{1,1} \right)^{-1}$$

Per computare il determinante di A si invertiranno in parallelo tutte le matrici $A[i]$.

Per invertire una matrice $A[i]$ si farà uso della seguente formula:

$$(I - xA[i])^{-1} = \sum_{j=0}^{\infty} (xA[i])^j$$

In particolare, essendo il determinante un polinomio di n-esimo grado, non sarà necessario iterare la sommatoria oltre il termine in cui la x avrà grado n.

Ottenute le inverse delle matrici, si moltiplicheranno gli elementi di coordinate $(1, 1)$ delle stesse.
Il polinomio così ottenuto avrà in generale un grado maggiore di n: se ne considereranno solo i termini in cui la variabile x è di grado non superiore ad n. Tali termini saranno scritti nella forma $c(1 + xp(x))$ (si assume per semplicità $c \neq 0$).

Invertendo,

$$c(1 + xp(x))^{-1} = c^{-1} \sum_{i=1}^{\infty} (-xp(x))^i \mod x^{n+1}$$

si ottiene $\det(I - xA)$; il coefficiente della n-esima potenza di x (moltiplicato per -1 se n è dispari) sarà il determinante di A.

Il tempo necessario è $\mathrm{O}(\log^2 n)$, il lavoro è $\mathrm{O}(n^4)$.

Riferimenti bibliografici

[P94] C. H. Papadimitriou, *Computational Complexity*, Addison-Wesley, 1994.

[CLRS01] T. H. Cormen, C. E. Leierson, R. L. Rivest, C. Stein, *Introduction to Algorithms, 2nd ed.*, MIT Press, 2001.