Implementazione distribuita del meccanismo Vickrey-Clarke-Groves

prof. ing. Paolo Santancini - paolo.santancini@studiosantancini.it

Punti del seminario

- Teoria dei giochi
- Algoritmi per la teoria dei giochi e progettazione di modelli
- “Social Choice”
- Modelli compatibili con incentivi
- Meccanismo VCG
- Zero-Knowledge Protocols
- La “cava digitale”
- Hands-on

Teoria dei giochi

- Studio di modelli matematici di interazioni strategiche tra agenti razionali
- Anni ‘30 primi risultati ottenuti su giochi a somma zero
- 1944 primo sviluppo e affermazione nel campo della matematica (*Von Neumann-Morgenstern “Theory of Games and Economic Behavior”*). Introduce l’idea di equilibri in strategie miste per giochi a somma zero e successivamente vengono considerati anche giochi cooperativi a più giocatori
- La teoria dei giochi moderna viene applicata maggiormente, e trova il suo continuo sviluppo, nell’economia, ingegneria e scienze sociali

Somma zero

- Interazione tra una coppia di concorrenti
- Ciò che un partecipante vince viene perso dall'altro. Ad esempio nel poker, se il giocatore A vince 100 euro contro il giocatore B, quest'ultimo perderà esattamente 100 euro (vincite - perdite = 0)

Giochi cooperativi

- Modellazione e analisi di comportamenti decisionali in situazioni di incertezza
- L'azione imprevista di un singolo può compromettere l'intera gara e volgerla esclusivamente a proprio vantaggio
- Esempio: mammuth o lepre?

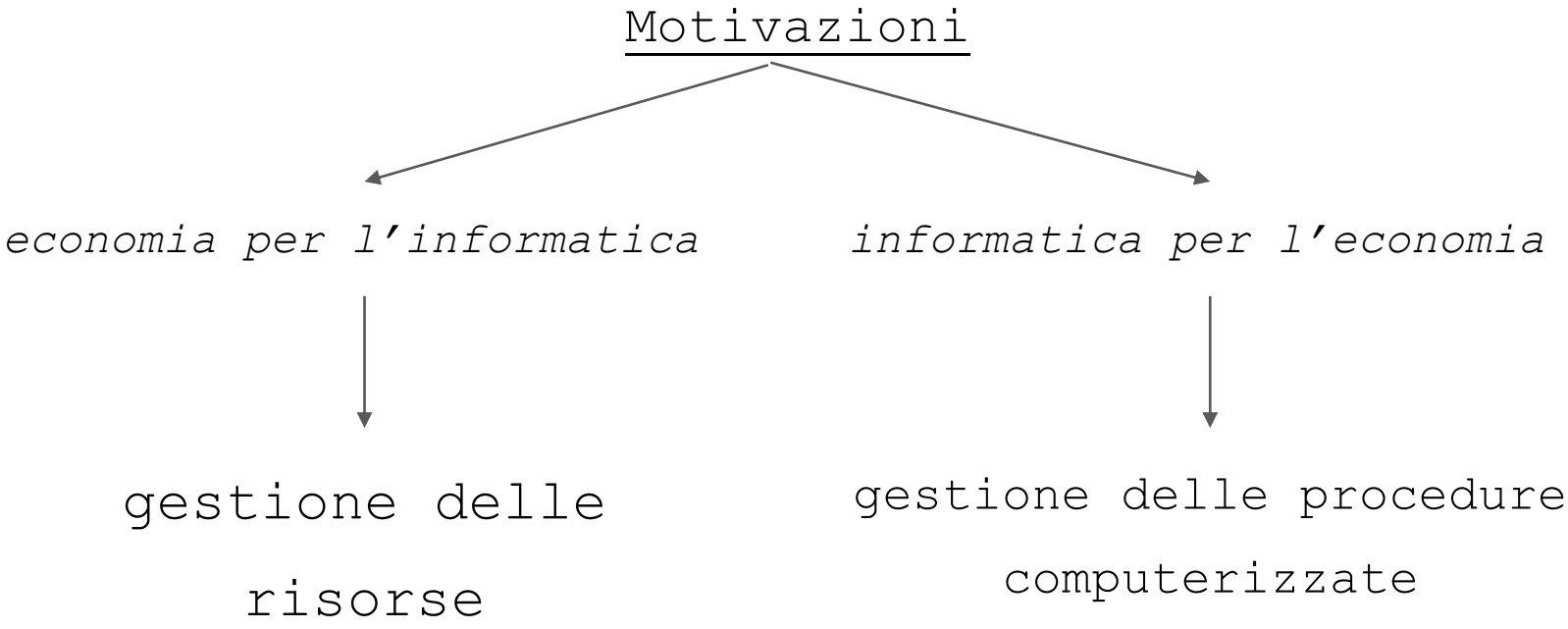
Equilibrio

- Ciascun giocatore deve seguire le proprie mosse migliorando la propria posizione in maniera non unilaterale
- Per cambiare, occorre agire insieme
- *John Nash*: «ciascun giocatore deve avere un profilo di strategie rispetto al quale nessun giocatore ha interesse ad essere l'unico a cambiare»

Algoritmi per la teoria dei giochi

- Area di intersezione tra teoria dei giochi e l'informatica. Ha come obiettivo il comprendere e progettare algoritmi in ambienti strategici
- Approccio ingegneristico e quantitativo
- Modellare applicazioni attraverso concreti problemi di ottimizzazione e ricerca soluzioni ottime
- Complessità polinomiale

Progettazione di modelli



Progettazione di modelli

- Algoritmi, protocolli, sistemi
- “Social choice”
  - elezioni
  - mercati
  - aste
  - politiche governative

Social Choice

- Aggregazione delle preferenze
- Raggiungere una condizione di equilibrio
- Collegare la scelta sociale a una determinata funzione che potrebbe o meno implicare a un paradosso:
  - “social welfare function”: l’insieme di tutte le preferenze per ciascun elemento da scegliere;
  - “social choice function”: l’insieme di tutte le scelte relative a un singolo selettore.

Social Choice - Condorcet

a Â$_1$ b Â$_1$ c; b Â$_2$ c Â$_2$ a; c Â$_3$ a Â$_3$ b

	Prima scelta	**Seconda scelta**	**Terza scelta**
Â$_1$	Scelta a	Scelta b	Scelta c
Â$_2$	Scelta b	Scelta c	Scelta a
Â$_3$	Scelta c	Scelta a	Scelta b

Social Choice - Condorcet

b $\hat{A}_1$ c; b $\hat{A}_2$ c; c $\hat{A}_3$ b

	Prima scelta	**Seconda scelta**
$\hat{A}_1$	Scelta b	Scelta c
$\hat{A}_2$	Scelta b	Scelta c
$\hat{A}_3$	Scelta c	Scelta b

Schwartz Sequential Dropping (SSD)

- first-past-the-post (paradosso);
- conteggio delle preferenze complessive tra i candidati (soluzione).

a Â1 b Â1 c Â1 d

a Â2 b Â2 d Â2 c

d Â3 c Â3 b Â3 a

b Â4 c Â4 d Â4 a

c Â5 b Â5 d Â5 a

Preferenze

Â1: 3 volte (a>b, a>c e a>d);

Â2: 3 volte (a>b, a>d e a>c);

Â3, Â4, Â5: 0 volte

	a	b	c	d
$\hat{A}_1$	3	2	1	0
$\hat{A}_2$	3	2	0	1
$\hat{A}_3$	0	1	2	3
$\hat{A}_4$	0	3	2	1
$\hat{A}_5$	0	2	3	1
	6	**10**	**8**	**6**

- "b" più gradito
- 5 volte seconda scelta, sopra la media 5 volte su 5

Modelli con incentivi

a Â$_i$ b ??? Quanto

- *A* (insieme delle sclete «a»), *I* (insieme dei partecipanti «i»);
- *m* peso (es. denaro);
- V_i funzione di valutazione;
- U_i funzione di preferenza quasi lineare;

$$U_i = V_i(a) + m$$

Second price auction

- Asta
- Competizione più equilibrata
- Reale valore percepito del prodotto
- Incentivo a non offrire il valore più alto
- Il vincitore paga la seconda offerta più alta più un peso

Second price auction

- *Esempio*: sistema di prenotazione di biglietti
  - N: numero biglietti
  - D: domanda (D >> N)
  - Pr: prezzo base del singolo biglietto (100€)
- *Anomalia*: richieste fuori prezzo base
  - P: prezzo finale del biglietto
  - f: «fee» (peso, maggiorazione) (5€)
- *Equilibrio?*

Second price auction

- Si procede con un’asta così strutturata:
  - Si crea una lista di offerte ordinata in modo decrescente
  - Si considerano i primi «n» valori
  - La seconda offerta maggiore, maggiorata di un peso, stabilirà il prezzo di ciascun biglietto tra i primi n partecipanti

Second price auction

Offerente	Valore
O1	144
O2	163
O3	134
O4	172
O5	153
O6	186
O7	158
O8	161

Offerente	Valore
O6	186
O4	172
O2	163
O8	161
O7	158
O5	153
O1	144
O3	134

Prezzo_finale = 172 + 5 = **177€**

Vickrey-Clarke-Groves

Peso	Scelta	Ads	€
10	1	C	8
8	2	D	7
3	3	A	5
1	4	B	1

1	2	3	4
80	64	24	8
70	56	21	7
50	40	15	5
10	8	3	1

$$A(v) \in \arg\max \sum_i v_i(a) \; ;$$

$$p_i(v) = \max \sum_k v_k(a) - \sum_k v_k\big(A(v)\big) \mid k \neq i$$

Vickrey-Clarke-Groves

Peso	Scelta	Ads	€
10	1	C	8
8	2	D	7
3	3	A	5
1	4	B	1

1	2	3	4
80	64	24	8
~~70~~	~~56~~	21	7
50	~~40~~	~~15~~	~~5~~
10	8	~~3~~	~~1~~

(70+40+3) - (56+15+1) = **41** Contributo Sociale **VCG_costo** = 41 + 70 = 111

Zero-knowledge Protocols

https://zkproof.org/

- *Instance*(x)
- *Witness*(w)
- *Relation* (R)
- *Language* (L)
- *Statement* (x ∈ L)
- *Security parameter* (k o s)
- *Setup*
  - setupP = (PrivateSetupP , CRS)
  - setupV = (PrivateSetupV , CRS)

ZKProof - Sintassi

- ***Prove***(stato, m) → (stato, p)
- ***Verify***(stato, p) → (stato, m)
- ***Setup***(params) → (setupR, setupP, setupV, auxi)

Piccolo Teorema di Fermat

se (p) è un numero primo allora per qualsiasi intero (a) elevato a (p) troviamo (a) come risultato della seguente equazione:

$$a^p \equiv a \pmod{p}$$

Piccolo Teorema di Fermat

a	p	a^p	a^p/p		a/p		
			INT	RESTO	INT	RESTO	
2	2	4	2	0	1	0	PRIMO
2	3	8	2	2	0	2	PRIMO
2	4	16	4	0	0	2	NON PRIMO
2	5	32	6	2	0	2	PRIMO
2	6	64	10	4	0	2	NON PRIMO
2	7	128	18	2	0	2	PRIMO
2	8	256	32	0	0	2	NON PRIMO
2	9	512	56	8	0	2	NON PRIMO
2	10	1024	102	4	0	2	NON PRIMO
2	14	16384	1170	4	0	2	NON PRIMO
2	15	32768	2184	8	0	2	NON PRIMO
2	16	65536	4096	0	0	2	NON PRIMO
2	17	131072	7710	2	0	2	PRIMO
2	18	262144	14563	10	0	2	NON PRIMO
2	19	524288	27594	2	0	2	PRIMO
2	20	1048576	52428	16	0	2	NON PRIMO
2	21	2097152	99864	8	0	2	NON PRIMO
2	22	4194304	190650	4	0	2	NON PRIMO

Rivest–Shamir–Adleman Cryptosystem

- Posti *p* e *q* due numeri primi molto grandi, con p ≠ q:

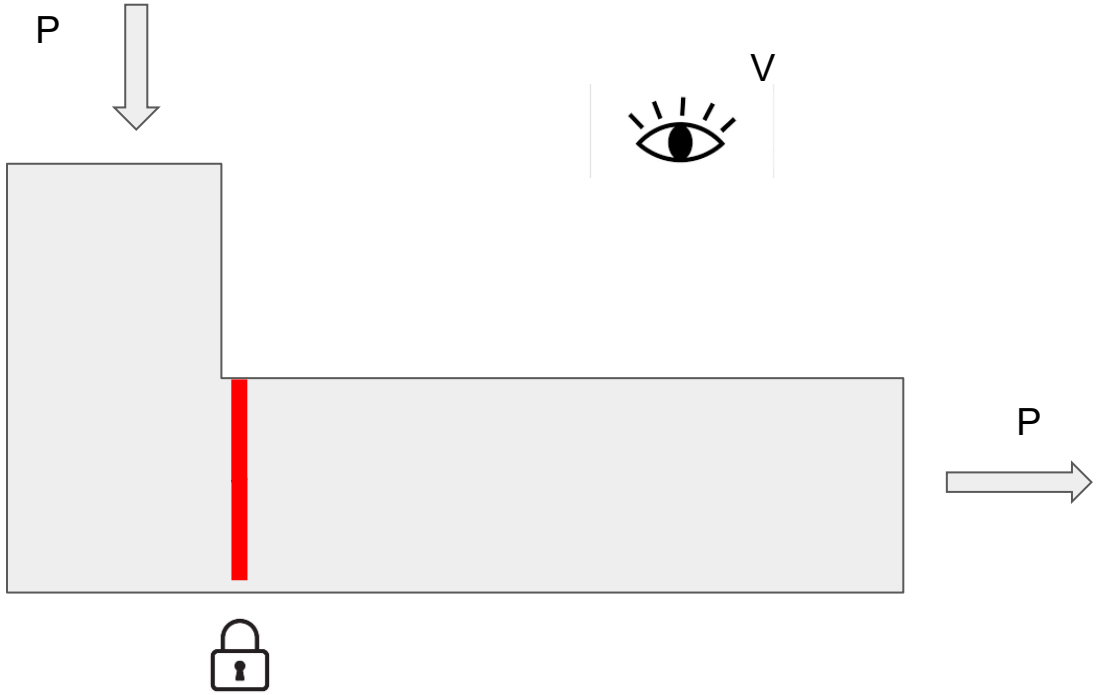
$$N = p * q$$
$$\varphi(N) = (p-1) * (q-1)$$

- Posto un numero $3 < e < \varphi(N)$, coprimo con $\varphi(N)$;
- La chiave primaria è il seguente risultato:

$e * d \equiv 1 \bmod (\varphi(N))$ | *d* è l'inverso moltiplicativo di *e*

- *(N,d) segreta; (N,e) pubblica.*

ZKProof - “La cava digitale”



ZKProof - “Esempio: risoluzione RSA”

Prover afferma di sapere che un messaggio “m” è criptato con RSA nel seguente modo:

$$m^e \equiv c \bmod (N)$$

con (N,c,e) ()=pubblico, [m] []=segreto

ZKProof - "Esempio: risoluzione RSA"

Verifier x1*x2 ≡ c mod (N)

dove

- x1 ≡ r1^e (mod N)
- x2 ≡ r2^e (mod N)
- r2 ≡ [m] * r1^(-1) (mod N) *(r1 -> random integer, r2, parametri conosciuti solamente dal Prover)*

ZKProof - "Non interattivo"

- *Prover* dimostra di conoscere [m] senza rivelarlo;
- *Verifier* non conosce [m];
- *Verifier* suppone che se il *Prover* conosce [m] (nascosto dal suo crittogramma c) può calcolare la funzione x1*x2 ≡ c mod (N);
- *Prover* dimostra di conoscere come fattorizzare (c), operazione per la quale trovare due numeri tali è problematica computazionalmente molto elevata.

Hands-on

- *Condizioni iniziali*
  - Posta una soglia m = 3 e un numero di share n = 5
  - Un *«prover»*, all'interno di una serie di competizioni distinte e pagando un «costo sociale», deve dimostrare di conoscere la sequenza per ricomporre una chiave segreta utile all'attivazione di una determinata risorsa (luce di una stanza)
  - Ciascun prover non conosce il comportamento degli altri concorrenti
  - I *«verifier», avendo in loro possesso e mettendo in gioco n-m share*, possono procedere alla verifica osservando la presenza di luce all'interno della stanza
  - Tra prover e verifier non deve aver luogo alcun trasferimento di informazioni «critiche»

- *Modello matematico – Shamir's treshold*
  - *m = soglia, n = share, S = segreto;*
  - *(m,n) | m < n;*

$$Y = a_0 + a_1 x + \cdots + a_{m-1} x^{m-1}$$

$$a_0 = S$$

$$Share_i = (x_i, y_i) \mid 0 < i \leqslant n$$

$$a_1, a_2, \ldots, a_{m-1} \mid interi > 0 \; casuali$$

Hands-on

$Y_a = 2023 + 7x + 3x^2$ curva del prover «a» con m = 3, n= 5

Gettone

Peso	(x_1, y_1)	Pr	g
8	(1,2033)	A	3
4	(1,789)	C	2
2	(1,1579)	B	1

Peso	(x_2, y_2)	Pr	g
5	(2,2049)	A	4
3	(2,1887)	C	2
1	(2,944)	B	1

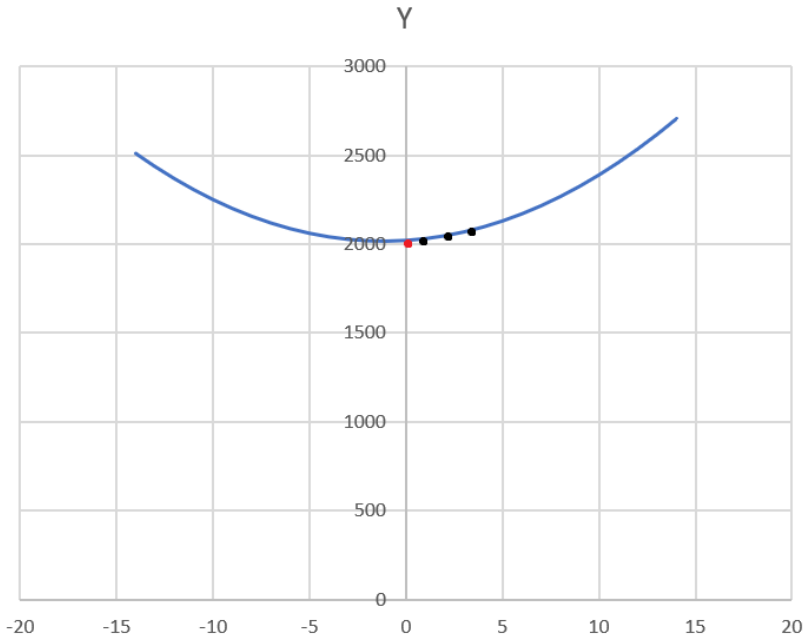
(1,2033)	(1,789)	(1,1579)
24	12	6
16	8	4
8	4	2

α

(2,2049)	(2,1887)	(2,944)
20	12	4
10	6	2
5	3	1

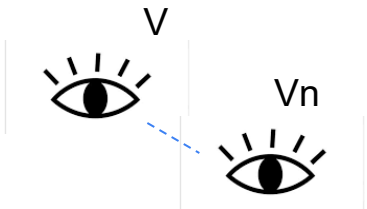
ß

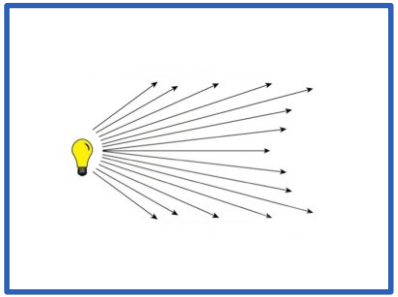
Hands-on



$p(\alpha) = 10 + 16$

$p(ß) = 6 + 10$





Hands-on

- *Come risultato il Prover «a»:*
  - dimostra di saper ricostruire il modello matematico necessario per raggiungere l'obiettivo
  - per ogni sessione paga un costo comprensivo del contributo sociale
  - Ottiene una precedenza sugli altri concorrenti
- In presenza di azioni inattese da parte degli altri partecipanti si deve ripetere la gara

Riferimenti

- ***https://github.com/paolosantancini/distributed_vcg***

- ***Algorithmic Game Theory***, Noam Nisan, Hebrew University of Jerusalem, Tim Roughgarden, Stanford University, California, Eva Tardos, Cornell University, New York, Vijay V. Vazirani, Georgia Institute of Technology - Cambridge University Press

- ***Cryptography Algorithms***, Massimo Bertaccini - Packt

Grazie per l'attenzione

prof. ing. Paolo Santancini – paolo.santancini@studiosantancini.it