POLITECNICO DI MILANO
Corso di Laurea in Ingegneria Informatica
Dipartimento di Elettronica e Informazione

UN APPROCCIO DI AUTOMATED MECHANISM DESIGN PER ASTE DI LINK SPONSORIZZATI CON MOTORI DI RICERCA FEDERATI

Relatore: Prof. Nicola Gatti
Correlatore: Ing. Sofia Ceppi

Tesi di Laurea di:
Davide Bacchini, matricola 711319

Anno Accademico 2009-2010

Sommario

I meccanismi d'asta per la visualizzazione di link sponsorizzati da parte di motori di ricerca stanno assumendo un'importanza sempre più rilevante all'interno della comunità scientifica. Gli inserzionisti offrono un certo valore per determinate parole e, quando l'utente effettua una ricerca, il motore utilizza un meccanismo d'asta per selezionare i link sponsorizzati da visualizzare a fianco dei risultati. Lo scopo della tesi è quello di realizzare un primo tentativo di estendere questo modello commerciale ad un nuovo paradigma di ricerca, chiamato Search Computing. Questo paradigma consiste nella realizzazione di un integratore che combini diversi motori di ricerca domain-specific federati. Esso permette infatti all'utente di formulare query multi-dominio che vengono automaticamente decomposte ed inviate ai relativi motori di ricerca specifici. L'integratore si preoccupa poi di combinare i risultati e presentarli all'utente. Proponiamo un modello commerciale per questo scenario, e sviluppiamo un meccanismo economico basato sulla teoria di automated mechanism design.

Indice

Capitolo 1

Introduzione

I meccanismi d'asta per la visualizzazione di link sponsorizzati da parte di motori di ricerca stanno assumendo un'importanza sempre più rilevante all'interno della comunità scientifica. La loro grande efficacia, unita al crescente utilizzo dei motori stessi, rende questa forma di pubblicità la più produttiva, in termini di ritorno economico, tra i diversi formati pubblicitari utilizzati su Internet. Lo scopo di questo lavoro di tesi consiste nello sviluppo di un modello economico di questo tipo, che possa essere adottato nell'ambito di un progetto innovativo per la realizzazione di un particolare motore di ricerca. Questo progetto, che prende il nome di Search Computing (SeCo), mira alla realizzazione di un meta-motore capace di combinare risultati provenienti da diversi motori domain-specific federati, al fine di migliorare l'efficienza e la precisione della ricerca.

1.1 Scenario applicativo

Il funzionamento di un asta per la visualizzazione di link sponsorizzati è piuttosto semplice. Per ogni ricerca effettuata dall'utente, il motore seleziona un certo numero di inserzionisti che vengono visualizzati nella pagina restituita, a fianco dei risultati veri e propri. Per scegliere quali inserzionisti visualizzare, il motore bandisce un'asta tra coloro che hanno dichiarato un certo interesse, specificando un'offerta, per la particolare parola ricercata. Se l'utente clicca su uno di questi link sponsorizzati, l'inserzionista paga una certa quantità al motore di ricerca. Questa quantità dipende dal meccanismo d'asta implementato dal particolare motore. Questi meccanismi sono stati estensivamente studiati nel corso degli ultimi anni, e funzionano sufficientemente bene per lo scenario appena descritto. Tuttavia, le continue

innovazioni in questo campo hanno portato alla nascita di nuovi paradigmi di ricerca, che richiedono la definizione di modelli appropriati. Uno di questi paradigmi è rappresentato dal progetto SeCo [17], il cui scopo consiste nella realizzazione di un meta-motore basato su un sistema di integrazione di motori di ricerca eterogenei. Questo modello permette all'utente di effettuare ricerche multi-dominio, grazie alla capacità di decomporre automaticamente una determinata query in più parti, ognuna relativa ad un singolo dominio, ed indirizzare ognuna di esse verso lo specifico motore di ricerca, combinando in seguito i risultati ottenuti. Nasce quindi l'idea di utilizzare, oltre ai risultati della ricerca, anche la lista di link sponsorizzati prodotta da ogni motore, al fine di realizzare un sistema pubblicitario basato anch'esso sull'integrazione. Precisiamo che meccanismi commerciali che prevedono l'utilizzo di motori di ricerca esterni, e la visualizzazione dei link sponsorizzati ad essi appartenenti, sono già presenti in letteratura e nelle applicazioni reali (ad esempio [7]). Essi potrebbero quindi essere scelti ed adattati anche per questo sistema. Riteniamo, tuttavia, che le particolari caratteristiche di questo paradigma richiedano un lavoro dedicato di ricerca allo scopo di fornire un sistema commerciale più adeguato allo scenario considerato. La formulazione di un modello economico ad hoc per questo innovativo sistema si rende quindi necessaria, ed è l'obiettivo di questo lavoro di tesi.

1.2 Il lavoro svolto

Vogliamo quindi proporre un meccanismo economico [18] che regoli l'interazione tra l'integratore SeCo e i motori di ricerca che esso utilizza, al fine di combinare in un'unica lista le diverse liste di link sponsorizzati provenienti da essi. Compito dell'integratore sarà quindi quello di decidere un sottoinsieme di inserzionisti tra quelli ricevuti, che verrà mostrato sulla propria pagina dei risultati. Questo processo di combinazione si basa sulla comunicazione da parte dei motori di ricerca dei valori di offerta e probabilità di click per ogni inserzionista. L'integratore deciderà quali inserzionisti visualizzare in modo da massimizzare una certa funzione obiettivo. La situazione descritta viene definita, all'interno della teoria dei giochi, come un gioco non cooperativo ad informazione incompleta. I valori di offerta e probabilità di click sono infatti informazioni private di ogni singolo motore di ricerca, ed è compito dell'integratore creare un meccanismo di incentivi che induca ogni agente in gioco a comunicarne i veri valori. Nel nostro lavoro, formuleremo questa situazione come un problema di mechanism design *one-stage* [13], e ne discuteremo le proprietà richieste. In particolare, per il processo di creazione del modello utilizzeremo un approccio di automated mechanism design [15],

che garantisce maggiore flessibilità nella modifica di vincoli e parametri del problema, oltre che nella scelta della funzione obiettivo. Tramite questo approccio andremo quindi a formalizzare quindi gli agenti in gioco, i vincoli da imporre per garantire le proprietà richieste e le funzioni obiettivo che il meccanismo vuole massimizzare. Presenteremo infine uno strumento automatico che genera, tramite il linguaggio di programmazione matematica AMPL, modelli di scenari casuali e pseudo-reali, dati alcuni parametri che l'utente definisce.

Concludiamo questa breve introduzione presentando i due principali vantaggi che a nostro avviso questo modello commerciale presenta. Innanzitutto, il processo di integrazione di link sponsorizzati permette di ottenere, come per quanto avviene con i contenuti, risultati più adeguati all'oggetto della ricerca, in quanto esso utilizza una maggiore quantità di informazioni provenienti da diverse fonti. In secondo luogo, questa innovativa soluzione fornisce la possibilità a piccoli motori di ricerca, specializzati in un singolo dominio, di combinarsi e completarsi diventando reali competitori dei più famosi motori di ricerca di carattere generale.

1.3 Struttura della tesi

La tesi è strutturata nel seguente modo. Il Capitolo 2 presenta lo stato dell'arte attuale, concentrandosi in particolare sulla teoria di mechanism design, sulle aste di link sponsorizzati e sul progetto Search Computing, ovvero sui tre ambiti in cui questo lavoro di tesi si colloca. Nel Capitolo 3 si formalizza la proposta innovativa da cui nasce la realizzazione del modello. Il Capitolo 4 rappresenta il cuore di questo lavoro di tesi, ovvero la descrizione matematica del modello formalizzato. Nel Capitolo 5 si presentano alcuni semplici casi di studio ed i primi risultati sperimentali. Si presenta inoltre una soluzione per la creazione automatica di modelli di scenari reali. Nel Capitolo 6, infine, si riassume il lavoro svolto traendo le dovute conclusioni, e si presentano le prospettive di sviluppo futuro, che per un modello ancora in fase di definizione come questo assumono senza dubbio una notevole importanza.

Capitolo 2

Stato dell'arte

Prima di proporre un modello commerciale per il sistema che abbiamo introdotto nel precedente capitolo, è necessario inquadrare più precisamente il problema. Abbiamo già accennato a come il sistema che vogliamo modellare possa essere rappresentato come un gioco tra agenti, che cercano di ottenere un profitto dal meccanismo. È necessario quindi fornire una definizione formale di questo gioco, e precisare come sia possibile stabilire delle regole che tutti i giocatori debbano seguire. Questi concetti vengono definiti dalla teoria di mechanism design, che presentiamo nella Sezione 2.1. Nella Sezione 2.2 formalizziamo invece il meccanismo alla base delle aste di link sponsorizzati. Concludiamo con una breve descrizione del progetto Search Computing, fornita nella Sezione 2.3.

2.1 Mechanism design

La teoria di mechanism design si colloca nell'ambito della risoluzione di problemi che riguardano sistemi multi-agente, nei quali ogni agente possiede informazioni private sconosciute agli altri. Essa è costituita da un insieme di strumenti matematici per definire le interazioni tra questi agenti, al fine di assicurare il raggiungimento di un esito desiderato. Vedremo come questi meccanismi rappresentino la soluzione naturale per descrivere il modello che andremo a considerare.

2.1.1 Teoria dei giochi

Ogni sistema multiagente è costituito da un insieme di agenti autonomi, che prendono decisioni sulla base delle informazioni a loro disposizione. Le decisioni vengono prese considerando le caratteristiche del sistema e le scelte

di tutti gli altri agenti in gioco. Questo tipo di problemi è molto comune in ambiti anche non prettamente informatici, come ad esempio l'economia o la biologia, e in alcuni casi la ricerca di una soluzione ottima può rivelarsi un problema di notevole importanza. Per questo motivo, nel corso degli anni è stato sviluppato un insieme di strumenti matematici per la descrizione di questi problemi. Questo insieme prende il nome di **teoria dei giochi** e fu introdotta per la prima volta nel 1944 in [14]. Diverse categorie di gioco sono state proposte e discusse all'interno della teoria dei giochi. Ci occuperemo ora di una particolare classe di giochi, ovvero quelli in *forma strategica* (o *normale*). Forniamone innanzitutto la definizione:

Def 2.1 (Gioco in forma strategica) *Un gioco in forma strategica* Γ *è definito da una tupla* $\langle N, (S_i)_{i \in N}, (u_i)_{i \in N} \rangle$*, dove* $N = 1, 2, ..., n$ *è un insieme finito di giocatori;* $S_1, S_2, ..., S_n$ *sono gli insiemi di strategie del giocatore* $1, 2, ..., n$ *rispettivamente;* $u_i : S_1 \times S_2 \times \ldots \times S_n \to \mathbb{R}$ *rappresentano le funzioni di utilità o di payoff.*

Le strategie sono anche chiamate azioni o strategie pure. Chiamiamo S il prodotto cartesiano $S_1 \times S_2 \times \ldots \times S_n$. Ogni profilo di strategie $s \in S$ induce un *esito* del gioco. L'utilità di ogni agente dipende da questo esito, ovvero dipende non solo dalla propria strategia, ma dalla strategia di tutti gli agenti in gioco. L'idea alla base dello studio dei giochi in forma strategica è quella di analizzare il problema decisionale da parte di ogni agente, nella scelta di un'azione che andrà a interagire con le decisioni di tutti gli altri agenti, producendo un esito. Concentriamoci quindi sul concetto di soluzione di un gioco strategico. Risolvere un gioco strategico consiste nell'individuarne alcune proprietà che consentano di stabilire, per ogni agente, quali siano le migliori strategie da adottare. Vediamo ora due tipi di soluzione molto comuni in letteratura: il primo utilizza il concetto di *strategia dominante*, mentre il secondo si basa sulla definizione di *equilibrio di Nash*. Esistono due nozioni di strategia dominante, si parla infatti di dominanza stretta e dominanza debole. Forniamo alcune definizioni:

Def 2.2 (Strategia strettamente dominata) *Sia dato un gioco*

$$\Gamma = \langle N, (S_i)_{i \in N}, (u_i)_{i \in N} \rangle$$

una strategia $s_i \in S_i$ *è detta strettamente dominata se esiste un'altra strategia* $s'_i \in S_i$ *tale che:*

$$u_i(s'_i, s_{-i}) > u_i(s_i, s_{-i}) \ \forall s_{-i} \in S_{-i}$$

Def 2.3 (Strategia strettamente dominante) *Una strategia $s_i^* \in S_i$ è detta strettamente dominante per il giocatore i se domina strettamente ogni altra strategia $s_i \in S_i$. Ovvero, $\forall s_i \neq s_i^*$:*

$$u_i(s_i^*, s_{-i}) > u_i(s_i, s_{-i}) \ \forall s_{-i} \in S_{-i}$$

Def 2.4 (Equilibrio in strategie strettamente dominanti) *Un profilo di strategie $(s_1^*, s_2^*, \ldots, s_n^*)$ è chiamato equilibrio in strategie strettamente dominanti per il gioco $\Gamma = \langle N, (S_i)_{i \in N}, (u_i)_{i \in N} \rangle$ se $\forall i = 1, 2, \ldots, n$ la strategia s_i^* è una strategia strettamente dominante per il giocatore i.*

Se in un gioco strategico un agente possiede una strategia strettamente dominante, ci si aspetta che esso la giochi sempre. Al contrario, ci si aspetta che una strategia strettamente dominata non sia mai giocata. Inoltre, se esiste un equilibrio in strategie strettamente dominanti, questo è unico.

Def 2.5 (Strategia debolmente dominata) *Sia dato un gioco*

$$\Gamma = \langle N, (S_i)_{i \in N}, (u_i)_{i \in N} \rangle$$

una strategia $s_i \in S_i$ è detta debolmente dominata se esiste un'altra strategia $s_i' \in S_i$ tale che:

$u_i(s_i', s_{-i}) \geq u_i(s_i, s_{-i}) \ \forall s_{-i} \in S_{-i}$ *e* $u_i(s_i', s_{-i}) > u_i(s_i, s_{-i})$ *per qualche* $s_{-i} \in S_{-i}$

Def 2.6 (Strategia debolmente dominante) *Una strategia $s_i^* \in S_i$ è detta debolmente dominante per il giocatore i se domina debolmente ogni altra strategia $s_i \in S_i$*

Def 2.7 (Equilibrio in strategie debolmente dominanti) *Un profilo di strategie $(s_1^*, s_2^*, \ldots, s_n^*)$ è chiamato equilibrio in strategie debolmente dominanti per il gioco $\Gamma = \langle N, (S_i)_{i \in N}, (u_i)_{i \in N} \rangle$ se $\forall i = 1, 2, \ldots, n$ la strategia s_i^* è una strategia debolmente dominante per il giocatore i*

A differenza di quanto accade per le strategie strettamente dominanti, possono esistere diverse strategie debolmente dominanti per ogni giocatore, e quindi diversi punti di equilibrio.

All'interno di un gioco strategico, queste due proprietà di equilibrio sono estremamente desiderabili, ma raramente soddisfacibili, in quanto necessitano di condizioni molto stringenti. Per questo motivo, è necessario introdurre la nozione di equilibrio di Nash, su cui si basa la classe di soluzioni più ampiamente usata all'interno della teoria dei giochi. Forniamo la definizione:

Def 2.8 (Equilibrio di Nash in strategie pure) *Sia dato un gioco*

$$\Gamma = \langle N, (S_i)_{i \in N}, (u_i)_{i \in N} \rangle$$

un profilo di strategie $(s_1^*, s_2^*, \ldots, s_n^*)$ *è un equilibrio di Nash in strategie pure se:*

$$u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*) \quad \forall s_i \in S_i \quad \forall i = 1, 2, \ldots, n$$

Un equilibrio di Nash è quindi un profilo di strategie in cui la strategia di ogni agente i è quella ottima se tutti gli agenti giocano la propria strategia di equilibrio. Se si dovesse quindi fornire un consiglio agli n agenti in gioco sulla strategia da seguire, questa sarebbe proprio quella indicata dall'equilibrio di Nash. Se invece dovesse accadere che uno o più agenti devino da questa strategia, è possibile che risulti più conveniente per un agente giocare in maniera differente.

Abbiamo definito alcune proprietà fondamentali che vanno ricercate in un gioco strategico. Vediamo ora un semplice esempio.

Esempio: Accordo su un divorzio Una coppia sta divorziando. Essi possiedono insieme un dipinto, e bisogna decidere cosa debba accadere a questo dipinto. Ognuno dei due ex-coniugi può mostrare un interesse alto o basso per il dipinto, e in base a questo verrà presa una decisione, secondo quanto mostrato in Tabella 2.1. Le decisioni possibili sono: il marito tiene il dipinto, la moglie tiene il dipinto, il dipinto viene donato ad un museo, oppure il dipinto viene bruciato. Ognuno dei due coniugi ha una certa preferenza sulle decisioni che possono essere prese, mostrata in Tabella 2.2, dove i valori in ogni casella rappresentano rispettivamente l'utilità della moglie e del marito. Abbiamo quindi un gioco in forma strategica definito da: due giocatori, due possibili azioni o strategie, quattro esiti e una funzione di utilità su questi esiti.

	marito_basso	marito_alto
moglie_basso	dipinto bruciato	dipinto al marito
moglie_alto	dipinto alla moglie	dipinto al museo

Tabella 2.1: Matrice degli esiti per il problema del divorzio

In questo semplice esempio troviamo un punto di equilibrio in strategie strettamente dominanti, determinato dalle azioni (alto,alto). Per entrambi gli agenti, infatti, la strategia migliore è sempre quella di giocare interesse alto, poichè i valori della funzione di utilità nel caso di strategia "alto" dominano

	marito_basso	marito_alto
moglie_basso	(0,0)	(0,5)
moglie_alto	(5,0)	(2,2)

Tabella 2.2: Matrice delle utilità per il problema del divorzio

quelli dell'altra strategia. Riprenderemo questo esempio nella Sezione 2.1.8, dove forniremo una diversa formulazione di questo problema.
Fino a questo punto, abbiamo parlato di giochi in forma strategica ad informazione completa, dove tutte le caratteristiche del gioco sono di comune conoscenza tra gli agenti. Nella realtà, tuttavia, molto spesso questa situazione non accade. Capita infatti che ogni singolo agente abbia a disposizione una certa quantità di informazioni che sono private, ovvero sconosciute agli altri giocatori. Queste informazioni prendono il nome di *tipi privati* dell'agente in questione. Si parla in questo caso di giochi ad informazione incompleta, che prendono anche il nome di giochi Bayesiani. Vediamo la definizione:

Def 2.9 (Gioco Bayesiano) *Un gioco Bayesiano* Γ *è definito da una tupla* $\langle N, (\Theta_i), (S_i), (p_i), (u_i)\rangle$*, dove* $N = 1, 2, ..., n$ *è un insieme finito di giocatori;* Θ_i *è l'insieme di tipi del giocatore* i*, con* $i = 1, 2, \ldots, n$*;* $S_1, S_2, ..., S_n$ *sono gli insiemi di strategie del giocatore* i*; la funzione di probabilità* p_i *è una funzione da* Θ_i *in* $\Delta(\Theta_{-i})$*, ovvero sull'insieme di distribuzioni di probabilità di* Θ_{-i}*.* $u_i : \Theta \times S \to \mathbb{R}$ *è una funzione di utilità, specificata per ogni profilo di azioni e ogni profilo di tipi.*

La struttura e le regole del gioco definite sono di comune conoscenza tra tutti gli agenti. L'unica informazione privata è rappresentata dal tipo di ogni singolo agente. Come per i giochi ad informazione completa, si rende necessaria la ricerca di punti di equilibrio. Forniamo la seguente definizione:

Def 2.10 (Equilibrio di Bayes-Nash) *Dato un gioco Bayesiano*

$$\langle N, (\Theta_i), (S_i), (p_i), (u_i)\rangle$$

un profilo di strategie

$$s^* = ((s^*_{\theta_1})_{\theta_1 \in \Theta_1}, (s^*_{\theta_2})_{\theta_2 \in \Theta_2}, \ldots, (s^*_{\theta_n})_{\theta_n \in \Theta_n})$$

è un equilibrio di Bayes-Nash in strategie pure se:

$$u_{\theta_i}(s^*_{\theta_i}, s^*_{-\theta_i}) \geq u_{\theta_i}(s_{\theta_i}, s^*_{-\theta_i}) \quad \forall s_i \in S_i \ \forall \theta_i \in \Theta_i \ \forall i \in N$$

Alternativamente, un profilo di strategie $(s_1^*(.), s_2^*(.), \ldots, s_n^*(.))$ *è un equilibrio di Bayes-Nash se:*

$$u_{\theta_i}(s_i^*(\theta_i), s_{-i}^*(\theta_{-i})) \geq u_{\theta_i}(s_i, s_{-i}^*(\theta_{-i})) \;\; \forall s_i \in S_i \, \forall \theta_i \in \Theta_i \, \forall \theta_{-i} \in \Theta_{-i} \, \forall i \in N$$

Vedremo come il gioco, che sta alla base del modello reale di cui questo lavoro di tesi si occupa, sia proprio di tipo Bayesiano, in quanto parte delle informazioni che gli agenti possiedono sono sconosciute agli altri. Queste definizioni ci torneranno quindi utili per lo studio delle proprietà richieste al modello.

Nella prossima sezione, vedremo come il problema della risoluzione di giochi strategici può essere affrontato tramite un metodo di reverse engineering. Questo approccio, infatti, fornisce uno strumento naturale per la risoluzione dei problemi stessi, che prende il nome di mechanism design.

2.1.2 Mechanism Design

Prima di addentrarci nelle caratteristiche fondamentali del mechanism design, partiamo da una sua definizione semplice e intuitiva: esso può essere visto come l'arte di progettare le "regole del gioco" in modo da ottenere un certo risultato desiderato. In un sistema multiagente, la teoria del mechanism design si pone quindi dal punto di vista di un *social planner*, che riceve le preferenze di ogni agente in gioco ed affronta il problema di prendere una decisione, sapendo tuttavia che le preferenze dichiarate di ogni agente possono non corrispondere a quelle vere. Per questo motivo, assumerà notevole importanza la creazione di adeguate regole atte ad incentivare gli agenti a comunicare le loro vere preferenze. A partire dalla seconda metà del ventesimo secolo, la teoria dei giochi ed il mechanism design hanno assunto un'importanza sempre più rilevante in un numero elevato di applicazioni ingegneristiche. Il mechanism design, in particolare, ha ottenuto un particolare successo in tempi molto recenti, col conferimento nel 2007 del premio Nobel per l'economia a Leonid Hurwicz, Eric Maskin, e Roger Myerson, per aver ideato i fondamenti di questa teoria. Un altro premio Nobel per l'economia era stato ricevuto nel 1996 da William Vickrey, inventore della famosa Vickrey auction. Questi premi dimostrano la crescente necessità di strumenti per modellare e risolvere problemi di questo tipo, dovuta alla significativa diffusione di sistemi in cui diversi agenti interagiscono strategicamente, in modo razionale ed intelligente.

2.1.3 Modello

Presentiamo ora una descrizione formale di un problema di mechanism design:

- Ci sono n agenti, $1, 2, ..., n$, con $N = 1, 2, ..., n$. Si assume che gli agenti siano razionali, ovvero che prendano sempre decisioni atte a massimizzare la propria funzione obiettivo. Sono supposti inoltre intelligenti, assumendo che essi abbiano a disposizione tutti gli strumenti matematici di teoria dei giochi e sufficiente capacità computazionale per determinare la propria strategia ottimale.
- Una scelta collettiva deve essere presa da un insieme di esiti X.
- Prima che la scelta collettiva venga presa, ogni agente valuta la sua preferenza sulle alternative di X. Questo viene modellato introducendo un parametro θ_i, sconosciuto agli altri agenti, che determina la sua preferenza. Esso viene chiamato *tipo privato* dell'agente i.
- Chiamiamo Θ_i l'insieme dei tipi privati dell'agente i. Un tipico profilo è definito da $\theta = \theta_1, ..., \theta_n$. L'insieme di tutti i possibili profili è $\Theta = \Theta_1 \times ... \times \Theta_n$.
- Esiste una certa distribuzione di probabilità $\Phi \in \Delta(\Theta)$ sull'insieme dei possibili profili, e da essa possono venire estratte tutte le informazioni che ogni agente possiede sugli altri giocatori.
- Le preferenze di ogni agente in gioco sono rappresentate da una funzione di utilità $u_i : X \times \Theta_i \to \mathbb{R}$. Dati $x \in X$ e $\theta_i \in \Theta_i$, il valore $u_i(x, \theta_i)$ rappresenta il payoff che l'agente i avente tipo θ_i riceve dalla decisione x.
- Il set di esiti X, il set di giocatori N, gli insiemi di tipi Θ_i, la distribuzione di probabilità Φ e le funzioni di utilità u_i sono assunte come conosciute da tutti gli agenti in gioco. L'unica informazione privata è il tipo θ_i di ogni agente.

Funzione di scelta sociale

Abbiamo finora parlato di una decisione collettiva che deve essere presa dal sistema. È naturale ipotizzare che essa dipenderà dal profilo θ, ovvero dalla realizzazione dei singoli tipi di ogni giocatore. Questa decisione prende il nome di funzione di **scelta sociale**, ed è definita come:

$$f : \Theta_1 \times ... \times \Theta_n \to X$$

La funzione di scelta sociale si preoccuperà quindi di assegnare ad ogni possibile profilo θ un elemento preso dall'insieme X dei possibili esiti.

Elicitazione e aggregazione delle preferenze

I tipi $\theta_1, ..., \theta_n$ di ogni singolo agente sono informazioni private dell'agente stesso, sconosciute quindi agli altri agenti e al social planner. Per questo motivo, data una funzione di scelta sociale $f : \Theta_1 \times ... \times \Theta_n \to X$, può succedere che un certo agente non sia interessato a comunicare il suo vero tipo. Si pone quindi un problema di *elicitazione delle preferenze*, che consiste nel creare un meccanismo che incentivi ogni agente a comunicare al decisore il vero tipo in cui esso si trova. Si deve fare in modo, quindi, che dire la verità rappresenti la strategia ottima per ogni agente in gioco.

Una volta garantito che ogni agente sia incentivato a riportare il proprio vero tipo, ci si pone il problema di fornire un risultato. Il decisore raccoglie quindi i tipi $\hat{\theta_i}$ riportati e fornisce un outcome basato sulla funzione di scelta sociale, ovvero calcola $f(\hat{\theta_1}, ..., \hat{\theta_n})$. Il cappello su $\hat{\theta_i}$ è per distinguere il tipo riportato dal vero tipo θ_i dell'agente i-esimo, che tuttavia coincide se il primo problema viene risolto. Il calcolo del risultato appena descritto è detto problema di *aggregazione delle preferenze*, ed è solitamente un problema di ottimizzazione.

Meccanismi diretti e indiretti

Abbiamo visto come il problema di mechanism design consista nel processo di soluzione di un problema di ottimizzazione non completamente specificato, attraverso l' elicitazione e l'aggregazione delle preferenze. Per specificare le informazioni sui tipi e le "regole del gioco", andremo ora a definire due tipi di meccanismi, ovvero meccanismi diretti e indiretti. Come detto prima, supponiamo che gli insiemi degli agenti N, degli esiti X, dei tipi $\Theta_1, ..., \Theta_n$, la distribuzione di probabilità Φ e le funzioni di utilità u_i siano conosciute da tutti gli agenti in gioco.

Data una funzione di scelta sociale $f : \Theta_1 \times ... \times \Theta_n \to X$, un meccanismo (di rivelazione) diretto consiste nella tupla $(\Theta_1, \Theta_2, ..., \Theta_n, f(.))$. L'idea di base è quella di estrarre *direttamente* le informazioni chiedendo agli agenti di rivelare il proprio vero tipo.

Un meccanismo (di rivelazione) indiretto consiste invece in una tupla $(S_1, S_2, ..., S_n, g(.))$, dove S_i è un insieme di possibili azioni che l'agente i-esimo può intraprendere e $g : S_1 \times S_2 \times ... \times S_n \to X$ è una funzione che assegna un outcome ad ogni profilo. L'idea è quella di introdurre un gioco

tra gli agenti specificato da azioni ed esiti sulle azioni stesse, la cui soluzione rivelerà *indirettamente* il tipo privato di ogni singolo agente.

2.1.4 Incentive compatibility

Abbiamo già accennato al problema di elicitazione delle preferenze, in cui vogliamo garantire che ogni agente riveli il proprio vero tipo. Definiamo ora quello che è probabilmente il concetto fondamentale all'interno della teoria di mechanism design, ovvero la nozione di *incentive compatibility (IC)*. Si definisce IC il fornire incentivi sufficienti tali da indurre tutti gli agenti in gioco a non mentire sul proprio tipo privato, partendo dal presupposto che ognuno di essi è intelligente e razionale.

Ci sono principalmente due tipi di IC: se comunicare la verità è la strategia ottima indipendentemente da quello che viene riportato dagli altri agenti si parla di *incentive compatibility in strategie dominanti* (DSIC); se invece comunicare la verità è la strategia ottima sapendo che ogni altro agente riporterà il proprio vero tipo si parla di *Bayesian-Nash incentive compatibility* (BIC). Forniremo ora le tre definizioni per un meccanismo di rivelazione diretto, in quanto esse fanno riferimento alla comunicazione dei tipi ed è quindi possibile definirle solo direttamente:

Def 2.11 (Incentive compatibility in strategie dominanti (DSIC)) *Una funzione di scelta sociale* $f : \Theta_1 \times ... \times \Theta_n \to X$ *è detta essere incentive compatible in strategie dominanti se il gioco Bayesiano indotto dal meccanismo di rivelazione diretto* $D = (\Theta_1, ..., \Theta_n, f(.))$ *ha un equilibrio debole in strategie dominanti* $s^*(.) = (s_1^*(.), ..., s_n^*(.))$*, dove* $s_i^*(\theta_i) = \theta_i, \forall \theta_i \in \Theta_i, \forall i \in N$

Def 2.12 (Bayesian incentive compatibility (BIC)) *Una funzione di scelta sociale* $f : \Theta_1 \times ... \times \Theta_n \to X$ *è detta essere Bayesian incentive compatible se il gioco Bayesiano indotto dal meccanismo di rivelazione diretto* $D = (\Theta_1, ..., \Theta_n, f(.))$ *ha un equilibrio di Bayes-Nash* $s^*(.) = (s_1^*(.), ..., s_n^*(.))$*, dove* $s_i^*(\theta_i) = \theta_i, \forall \theta_i \in \Theta_i, \forall i \in N$

2.1.5 Revelation principle

Se l'*incentive compatibility* rappresentava il concetto fondamentale all'interno della teoria di Mechanism Design, il *revelation principle* può probabilmente essere considerato il risultato più significativo della teoria stessa. Esso descrive la relazione che sussiste tra un meccanismo di rivelazione indiretto M ed uno diretto D, data una certa funzione di scelta sociale $f(.)$. Questo

risultato permette di restringere la ricerca di un'implementazione veritiera della funzione $f(.)$ alla sola classe di meccanismi di rivelazione diretti.

Th 2.1 (RP per DSIC) *Se esiste un meccanismo* $M = (S_1, S_2, ..., S_n, g(.))$ *che implementa la funzione di scelta sociale f(.) in strategie dominanti, allora f(.) è DSIC.*

Th 2.2 (RP per BIC) *Se esiste un meccanismo* $M = (S_1, S_2, ..., S_n, g(.))$ *che implementa la funzione di scelta sociale f(.) in strategie Bayesian-Nash, allora f(.) è BIC.*

2.1.6 Proprietà della funzione di scelta sociale

Vediamo ora due importanti proprietà della funzione di scelta sociale che vogliamo ottenere dalla definizione delle regole del gioco, ovvero le proprietà di individual rationality (IR) e di efficienza. La prima implica che ogni agente riceva un'utilità non negativa partecipando al meccanismo che implementi la funzione di scelta sociale. Sostanzialmente questa proprietà vuole garantire che ogni agente sia incentivato a giocare. Ci sono tre stadi in cui questa proprietà può essere garantita: dopo che i tipi sono stati comunicati e l'outcome x è stato scelto (*ex-post IR*), dopo che ogni agente abbia appreso il proprio tipo ma prima che abbia intrapreso una certa azione (*ex-interim IR*) oppure prima che ogni agente abbia appreso il proprio tipo (*ex-ante IR*). Per ogni SCF $f(.)$ abbiamo che:

$$f(.) \text{ is ex-post } IR \Rightarrow f(.) \text{ is ex-interim } IR \Rightarrow f(.) \text{ is ex-ante } IR$$

Vediamo nel dettaglio la proprietà che in seguito richiederemo nel nostro modello, ovvero quella di *ex-post IR*. Chiamiamo $\bar{u}_i(\theta_i)$ l'utilità che l'agente i riceve se si ritira dal meccanismo d'asta quando il suo tipo privato è θ_i. Il vincolo risulta:

$$u_i(f(\theta_i, \theta_{-i}), \theta_i) \geq \bar{u_i}(\theta_i) \ \forall(\theta_i, \theta_{-i}) \in \Theta$$

Allo stesso modo possiamo definire la proprietà di efficienza ex-post, interim ed ex-ante. Anche in questo caso ne forniamo la definizione *ex-post*: essa garantisce che per ogni profilo $\theta \in \Theta$ l'esito $f(\theta)$ sia Pareto-ottimale. L'esito $f(\theta_1, \ldots, \theta_i)$ è Pareto ottimale se non esiste un esito $x \in X$ tale che:

$u_i(x, \theta_i) \geq u_i(f(\theta), \theta_i) \ \forall i \in N \ and \ u_i(x, \theta_i) > u_i(f(\theta), \theta_i)$ per qualche $i \in N$

2.1.7 Introduzione pagamenti

Abbiamo visto in precedenza che una delle migliori proprietà da ricercare per una funzione di scelta sociale è quella di *incentive compatibility in strategie dominanti (DSIC)*. Tuttavia, esiste in letteratura un teorema di notevole importanza all'interno della teoria del mechanism design, che descrive come il raggiungimento di questa proprietà precluda il soddisfacimento di altre proprietà altrettanto desiderabili. Stiamo parlando del teorema di impossibilità di Gibbard-Satterthwaite [2, 16]. Esso in breve dimostra come la proprietà DSIC costringa la funzione di scelta sociale ad essere dittatoriale se lo spazio dell'utilità è non-ristretto. Una funzione sociale $f : \Theta \to X$ è detta dittatoriale se esiste un agente d (chiamato dittatore) che soddisfa la seguente proprietà:

$$\forall \theta \in \Theta, f(\theta) \text{ è tale che } u_d(f(\theta), \theta_d) \geq u_d(x, \theta_d) \forall x \in X$$

Ovvero, ogni outcome che può essere scelto dalla funzione di scelta sociale è anche outcome ottimale per il dittatore. Questa condizione è ovviamente da evitare per un sistema in cui si vogliono garantire pari opportunità a tutti gli agenti in gioco. Per questo motivo, per il nostro modello utilizzeremo una particolare classe di spazio per cui questo teorema non vale, ovvero lo spazio dell'utilità quasi-lineare. In questo sistema, ogni alternativa $x \in X$ è un vettore della forma $x = (k, t_1, ..., t_n)$, dove k è un elemento del set di allocazioni K, supposto finito. I termini $t_i \in \mathbb{R}$ rappresentano un trasferimento monetario dall'agente i (se $t_i < 0$) oppure verso l'agente stesso (se $t_i > 0$). Il set di alternative è definito da:

$$X = (k, t_1, ..., t_n) : k \in K; t_i \in \mathbb{R} \ \forall i \in N; \sum_i t_i \leq 0$$

Vediamo ora due importanti proprietà della funzione di scelta sociale in spazio dell'utilità quasi-lineare, ovvero le proprietà di *allocative efficiency* e di *budget balance*:

Def 2.13 (Allocative Efficiency (AE)) *Una funzione di scelta sociale* $f(.) = (k(.), t_1(.), ..., t_n(.))$ *è efficiente allocativamente se, per ogni* $\theta \in \Theta$, $k(\theta)$ *soddisfa la seguente condizione:*

$$k(\theta) \in \arg\max_{k \in K} \sum_{i=1}^{n} v_i(k, \theta_i).$$

Def 2.14 (Budget Balance (BB)) *Una funzione di scelta sociale* $f(.) = (k(.), t_1(.), ..., t_n(.))$ *è budget balanced se, per ogni* $\theta \in \Theta$, $t_1(\theta), ..., t_n(\theta)$

soddisfa la seguente condizione:

$$\sum_{i=1}^{n} t_i(\theta) = 0.$$

Questa proprietà è anche chiamata di *strong budget balance*, per distinguerla dalla condizione più debole $\sum_{i=1}^{n} t_i(\theta) \leq 0$, chiamata *weak budget balance.*

La relazione tra queste proprietà e la proprietà di ex-post efficiency della funzione di scelta sociale è descritta dal seguente lemma:

Lemma 2.1 *Una funzione di scelta sociale* $f(.) = (k(.), t_1(.), ..., t_n(.))$ *è ex-post efficiente in uno spazio dell'utilità quasi lineare se e solo se è allocativamente efficiente e budget balanced.*

Enunciamo ora un'altra importante caratteristica di questo ambiente:

Lemma 2.2 *Tutte le funzioni di scelta sociale sono non dittatoriali in uno spazio dell'utilità quasi lineare.*

Abbiamo visto che, se ci poniamo in uno spazio dell'utilità quasi-lineare, esistono funzioni di scelta sociale che sono sia allocativamente efficienti sia incentive compatible in strategie dominanti. Una classe di funzioni che rispetta queste proprietà prende il nome di meccanismi VCG (Vickrey-Clarke-Groves).

2.1.8 Automated mechanism design

Abbiamo descritto come il mechanism design sia l'arte di progettare le regole del gioco in modo da raggiungere un risultato desiderato, sebbene gli agenti si comportino egoisticamente. Questa teoria sta assumendo un interesse sempre più crescente all'interno della comunità scientifica, in particolare nell'ambito dell'Intelligenza Artificiale. Tuttavia, in molti casi di studio, i classici meccanismi presenti in letteratura non risultano essere soddisfacenti. I motivi possono essere molteplici, ad esempio in molti casi essi richiedono assunzioni troppo stringenti sulle azioni degli agenti, oppure ricercano l'obiettivo sbagliato, oppure non fanno uso di tutte le informazioni a loro disposizione. Per questo motivo, in [15] Tuomas Sandholm propone un approccio diverso per la risoluzione di questi problemi, chiamato *automated mechanism design (AMD).* La sua idea è quella di risolvere un problema di mechanism design trasformandolo in un problema di ottimizzazione. Il modello viene formalizzato nel seguente modo, sono dati:

- Un insieme finito di outcome O (in alcuni casi associati ad una funzione di costo su di essi).

- un insieme finito di N agenti.
- Per ogni agente i: un insieme finito di tipi Θ_i, una distribuzione di probabilità γ_i su Θ_i e una funzione di utilità $u_i : \Theta_i \times O \to \mathbb{R}$.
- Una funzione obiettivo che il creatore del meccanismo vuole massimizzare.
- Una specifica di quali strumenti sono a disposizione del creatore (ad esempio se sono possibili i pagamenti, la randomizzazione, ecc.).

I vantaggi che questa soluzione apporta sono molteplici. Innanzitutto, essa si appoggia a un software che viene creato una sola volta e che può essere applicato ogni volta che un nuovo problema sorge. Quindi la complessità di progettazione di un meccanismo appropriato viene trasferita dall'uomo alla macchina. La formulazione come problema di ottimizzazione può inoltre portare a un risultato migliore dell'obiettivo che il creatore del meccanismo vuole massimizzare. Un altro vantaggio è che spesso questo approccio porta ad evitare alcuni risultati di impossibilità presenti in letteratura, o almeno minimizzare le difficoltà che essi implicano. Essendo un approccio automatico, inserire nuove caratteristiche o ricercare funzioni obiettivo non convenzionali risulta essere semplice da implementare. Infine, in molti casi questo meccanismo porta a rinforzare le proprietà di incentive compatibilty e di partecipazione tra agenti, favorendo una più efficiente massimizzazione dei risultati ottenuti.

Riprendiamo ora l'esempio trattato nella sezione 2.1.1, e ne forniamo la formulazione in AMD tratta da [5].

Accordo su un divorzio in formulazione AMD

Consideriamo di nuovo il problema di assegnazione di un dipinto, appartenente a due coniugi che stanno divorziando. Abbiamo detto che ognuno dei due ex-coniugi può mostrare un alto o un basso interesse per il dipinto. Questo interesse, che nella formulazione del problema come gioco strategico rappresentava un'azione da parte degli agenti, rappresenta ora il tipo privato di ognuno dei due coniugi. Introduciamo una probabilità su questo tipo, assumendo che ogni coniuge mostri interesse alto con probabilità 0.8 e interesse basso con probabilità 0.2. Ricordiamo i quattro possibili esiti: il marito tiene il dipinto, la moglie tiene il dipinto, il dipinto viene donato ad un museo, il dipinto viene bruciato. Questa volta, la funzione di utilità è definita non solo sugli esiti, ma dipende anche dal tipo privato in cui ogni agente si trova. Assumiamo che sia la seguente:

- u(basso,ottengo il dipinto) = 2
- u(basso,il coniuge ottiene il dipinto) = 0
- u(basso,il dipinto viene donato al museo) = 1
- u(basso,dipinto bruciato) = -10
- u(alto,ottengo il dipinto) = 100
- u(alto,il coniuge ottiene il dipinto) = 0
- u(alto,il dipinto viene donato al museo) = 50
- u(alto,dipinto bruciato) = -10

Un'ulteriore differenza è rappresentata dalla presenza di un arbitro. Gli esiti, infatti, non sono più determinati unicamente dalle azioni che gli agenti compiono, ma è invece compito dell'arbitro quello di prendere una decisione, basata sull'informazione che essi comunicano. Per incentivare gli agenti a comunicare il proprio vero tipo, ovvero garantire la proprietà di incentive compatibility vista in precedenza, esso può introdurre dei pagamenti, che possono essere ricevuti o effettuati dai diversi giocatori. La scelta di quale decisione sia meglio prendere si basa sulla massimizzazione di una certa funzione obiettivo. L'arbitro può scegliere, ad esempio, di massimizzare i pagamenti da esso ricevuti, oppure l'utile complessivo atteso, oppure ancora l'utile di un singolo agente in gioco. Rimandiamo alla bibliografia per la completa esposizione dei risultati con diversi settaggi e diverse funzioni obiettivo. Vediamo solo il risultato che più si avvicina al problema di cui questa tesi tratta. Ci poniamo nel caso di un arbitro che possa ricevere pagamenti dagli agenti, il cui obiettivo sia quello di massimizzare questi pagamenti. Ovviamente, esso non può prelevare arbitrariamente la somma che desidera da ogni agente, ma deve prendere decisioni e comportarsi in modo da incentivare gli agenti a partecipare al gioco (vincolo di individual rationality). Implementando il gioco in strategie dominanti, con il vincolo di ex-post IR, il risultato che otteniamo è:

	marito_basso	marito_alto
moglie_basso	dipinto bruciato	dipinto al marito
moglie_alto	dipinto alla moglie	dipinto alla moglie

Notiamo subito come in uno dei casi la decisione migliore da prendere per l'arbitro sia quella di bruciare il dipinto, sebbene ogni agente ne ricavi un'utilità negativa e l'arbitro non ne ricavi nessun guadagno. Questo deriva dal

meccanismo fondamentale di incentivi che sta alla base della risoluzione del problema, e mostra come la soluzione che massimizza la funzione obiettivo possa essere, in alcuni casi, controintuitiva. La teoria dell'AMD ci garantisce però che proprio questa sia la decisione che massimizza il guadagno complessivo dell'arbitro, garantendo allo stesso modo che gli agenti siano incentivati a giocare.

Abbiamo trattato questo semplice esempio per motivare la scelta di un meccanismo di AMD per il nostro lavoro. Già per un semplice problema come quello esposto, infatti, un approccio di mechanism design classico (tipicamente un processo manuale) garantirebbe poca flessibilità. I due più famosi meccanismi in letteratura, ovvero VCG e dAGVA [13], permettono di massimizzare unicamente il social welfare. Meccanismi che massimizzano i guadagni sono conosciuti unicamente sotto requisiti molto stringenti (Myerson's o Maskin e Riley's expected revenue maximizing auction [13]). Con questo approccio, invece, è possibile di volta in volta creare automaticamente un meccanismo ad hoc per lo specifico problema da risolvere, e per la specifica funzione obiettivo che vorremo andare a massimizzare. La descrizione del modello formale basata sull'AMD verrà presentata nel Capitolo 4.

2.2 Sponsored search auctions

Ogni volta che un utente inserisce una parola in un motore di ricerca, il sistema gli restituisce una pagina di risultati. In questa pagina, oltre ai risultati veri e propri della ricerca, sono presenti alcuni link sponsorizzati, collocati ad hoc in alcune posizioni della pagina web. Se l'utente clicca su uno di questi link, viene trasferito sulla pagina dell'inserzionista, che per questo sevizio paga una certa quantità al motore di ricerca. Trattandosi di link pubblicitari, lo spazio in cui il motore di ricerca può posizionare questi inserzionisti è limitato ad un certo numero di posizioni disponibili. Per questo motivo, ad ogni ricerca effettuata dall'utente, il motore deve affrontare il problema di scegliere quali inserzionisti mostrare in queste posizioni. Inoltre, esso deve determinare alcuni criteri per stabilire la quantità che gli inserzionisti cliccati devono pagare. Per l'allocazione delle posizioni disponibili, la maggior parte dei motori di ricerca utilizza un meccanismo d'asta. Questi meccanismi prendono il nome di Sponsored Search Auctions, e funzionano nel seguente modo: ogni inserzionista specifica un certo insieme di parole a cui è interessato, assieme a un'offerta relativa a ognuna di queste parole. In base ai valori di queste offerte, a ogni ricerca il motore sceglie un sottoinsieme di inserzionisti tra quelli interessati alla particolare parola ricercata. Il prezzo che ogni inserzionista paga per il click dell'utente

dipende anche dal valore di queste offerte. Nelle prossime sezioni, andremo a formalizzare il modello descritto.

2.2.1 Modello formale

Consideriamo un motore di ricerca che ha ricevuto una query da un utente. Esso deve condurre un asta per stabilire come distribuire il proprio spazio pubblicitario in relazione alla parola cercata. Nella descrizione del modello facciamo le seguenti assunzioni:

- Ci sono n inserzionisti interessati a una particolare parola e $N = 1,2,...,n$ rappresenta l'insieme di questi inserzionisti. Ci sono inoltre m posizioni disponibili dove essi possono essere visualizzati, definite dall'insieme $M = 1,2,...,m$.

- α_{ij} è la probabilità che l'utente clicchi l'inserzionista i-esimo se questo viene mostrato nella posizione j. Essa quindi non dipende dagli inserzionisti mostrati nelle altre posizioni (*absence of allocative externality*). Assumiamo inoltre che:

$$1 \geq \alpha_{i1} \geq \alpha_{i2} \geq ... \geq \alpha_{im} \geq 0 \ \forall i \in N$$

- L'inserzionista i conosce con esattezza il valore che deriva dal click di un utente su di un proprio link, ma ignora quanto valga questo valore per gli altri inserzionisti. Viene definito il parametro θ_i per rappresentare questo valore, ed esso prende il nome di *tipo* dell'inserzionista i. L'insieme dei possibili tipi è definito da Θ_i.

- Ogni inserzionista percepisce le valutazioni degli altri come se fossero estratte da una certa distribuzione di probabilità, e sa che gli altri agenti descrivono allo stesso modo la sua. Formalmente, il comportamento di ogni inserzionista i è definito da una certa distribuzione di probabilità Φ_i da cui esso estrae il proprio tipo θ_i. Assumiamo che i valori possibili siano compresi in un intervallo chiuso $\left\lfloor \theta_i^{min}, \theta_i^{max} \right\rceil$. Inoltre le distribuzioni $\Phi_i(.), i = 1, 2, ..., n$ sono mutualmente indipendenti, ovvero la valutazione di ogni singolo agente non dipende dagli altri agenti (*indipendent private values assumption*).

- Ogni inserzionista è intelligente e razionale, e cerca di massimizzare una certa funzione di utilità $u_i : X \times \Theta_i \to \mathbb{R}$.

- L'insieme dei tipi $\Theta_1, ..., \Theta_n$, le distribuzioni di probabilità $\Phi_i(.)$ e le funzioni di utilità $u_i(.)$ sono conosciute da ogni agente in gioco. Questo

implica che ogni inserzionista conosca la funzione di utilità per ogni altro inserzionista j associata ad ogni suo tipo privato θ_j.

Possiamo ora formalizzare il modello di una sponsored search auction. Per ogni parola che può essere ricercata, ogni inserzionista i interessato offre una certa quantità $b_i \geq 0$, che dipende dal proprio tipo privato θ_i. Ogni volta che il motore di ricerca riceve una richiesta per questa parola, estrae dal proprio database le informazioni sugli n inserzionisti interessati e il corrispondente vettore di offerte $b = (b_1, ..., b_n)$. A questo punto, decide gli inserzionisti vincenti e in quale ordine essi verranno visualizzati (*regola di allocazione*). Stabilisce inoltre quanto ogni inserzionista debba pagare se l'utente clicca sul link corrispondente (*regola di pagamento*).

Una volta formulato il problema in questo modo, risulta naturale descriverlo come un problema di mechanism design. Formalmente, una *sponsored search auction* può essere vista come un meccanismo indiretto $M = ((B_i)_{i \in N}, g(.))$, dove $B_i \subset \mathbb{R}^+$ è l'insieme di tutte le possibili offerte dell'inserzionista i e $g(.)$ è la funzione di allocazione e pagamento. Se inoltre assumiamo che, per ogni inserzionista i, l'insieme delle offerte B_i coincida con il tipo Θ_i, il meccanismo indiretto $M = ((B_i)_{i \in N}, g(.))$ diventa un meccanismo di rivelazione diretto $D = ((\Theta_i)_{i \in N}, f(.))$, dove $f(.)$ è la regola di allocazione e pagamento. D'ora in poi, assumeremo che $B_i = \Theta_i \ \forall i = 1, .., n$ e descriveremo quindi il meccanismo di sponsored search auction come un meccanismo di rivelazione diretto.

Esiti X L'esito di una sponsored search auction viene rappresentato con un vettore $x = (y_{ij}, p_i)_{i \in N, j \in M}$, dove y_{ij} è la probabilità che l'inserzionista i sia posizionato nello slot j, e p_i è il prezzo che esso deve pagare per click dell'utente. L'insieme delle possibili alternative è definito da:

$$X = \left\{ (y_{ij,p_i})_{i \in N, j \in M} \mid y_{ij} \in [0,1] \ \forall i \in N, \forall j \in M, \sum_{i=1}^{n} y_{ij} \leq 1 \ \forall j \in M, \sum_{j=1}^{m} y_{ij} \leq 1 \ \forall i \in N \right\}$$

Funzione di utilità Dato $x = (y_{ij}, p_i)_{i \in N, j \in M}$, la funzione di utilità per l'inserzionista i è data da:

$$u_i(x, \theta_i) = \left(\sum_{j=1}^{m} y_{ij} \alpha_{ij} \right) (\theta_i - p_i)$$

Funzione di scelta sociale La struttura generale per le regole di allocazione e pagamento è definita come:

$$f(b) = (y_{ij}(b), p_i(b))_{i \in N, j \in M}$$

dove $b = (b_1, ..., b_n)$ è il vettore delle offerte, $y_{ij}(.)$ la regola di allocazione e $p_i(.)$ la regola di pagamento.

Linearizzazione Con una leggera modifica delle regole di allocazione e di pagamento e delle funzioni di utilità è possibile fornire una formulazione lineare del meccanismo di rivelazione diretto. Ridefiniamo quindi le regole come:

$$f(b) = (y(b), t_i(b))_{i \in N, j \in M}$$

dove $y(b) = (y_{ij}(b))_{i \in N, j \in M}$ e $t_i(b) = (\sum_{j=1}^{m} y_{ij}(b)\alpha_{ij})p_i(b)$. La variabile $t_i(b)$ può essere interpretata come la quantità media pagata dall'inserzionista i al motore di ricerca per ogni ricerca effettuata.

Ridefiniamo inoltre le funzioni di utilità come:

$$u_i(f(b), \theta_i) = \theta_i v_i(y(b)) - t_i(b)$$

dove $v_i(y(b)) = (\sum_{j=1}^{m} y_{ij}(b)\alpha_{ij})$. La quantità $v_i(y(b))$ può essere interpretata come la probabilità che l'inserzionista i venga cliccato ad ogni ricerca del motore.

Il modello di una sponsored search auction come meccanismo di rivelazione diretto è ora completamente definito. Andiamo ora a descrivere brevemente i due meccanismi ad oggi più utilizzati per la definizione delle regole di allocazione e pagamento, ovvero il *Generalized Second Price mechanism (GSP)* e il *Vickrey-Clarke-Groves mechanism*. Per una trattazione più completa si rimanda a [13].

2.2.2 Generalized Second Price mechanism

Il primo meccanismo d'asta utilizzato per la ricerca sponsorizzata è stato il cosiddetto Generalized First Price (GFP), introdotto da GoTo nel 1992. Tuttavia, questo meccanismo presentava una grande instabilità, dovuta principalmente al fatto che la comunicazione della verità non rappresenta una strategia di equilibrio per gli inserzionisti. Per ovviare a questo problema, nel 2002 Google introdusse il suo nuovo programma *AdWord Select*, basato sul meccanismo Generalized Second Price (GSP). Riconosciuti gli evidenti vantaggi, i principali motori di ricerca, tra i quali Yahoo!, si adattarono a questa soluzione. Vediamo ora le principali caratteristiche di questo meccanismo.

Regola di allocazione

Descriviamo tre regole di allocazione basate sul meccanismo GSP:

- **Yahoo!/Overture Allocation Rule** Le m posizioni disponibili sono allocate agli inserzionisti in base al valore della loro offerta b_i. Dato $b = b_1, \ldots, b_n$, vettore delle offerte ricevute ordinato decrescentemente, la prima posizione andrà all'inserzionista con offerta b_1, il secondo a quello con b_2, e così via.

- **Greedy Allocation Rule** Cerca di massimizzare la *allocative efficiency*. La prima posizione viene assegnata all'inserzionista i per cui il valore $\alpha_{i1}b_i$ è massimo. Una volta rimosso, si assegna la seconda posizione all'inserzionista con $\alpha_{i2}b_i$ massimo e si prosegue in questo modo per tutti gli slot.

- **Google's Allocation Rule** Deriva dalla regola greedy e si basa sul calcolo del *Click-Through-Rate* per ogni advertiser, ovvero il rapporto tra il numero di click ricevuti da un inserzionista e il numero di volte che esso è stato visualizzato. Gli inserzionisti vengono ordinati in base al proprio *ranking score*, ovvero il prodotto tra la propria offerta e il CTR.

Relazione tra probabilità di click e CTR

Vediamo ora la relazione tra le due quantità che abbiamo appena nominato, ovvero tra la probabilità di click e il Click-Through-Rate. Ricordiamo che:

- α_{ij} è la probabilità che l'utente clicchi l'advertiser i-esimo se questo viene mostrato nella posizione j.

- CTR_i è la probabilità che l'utente i clicchi l'advertiser i-esimo se questo viene mostrato.

- y_{ij} è la probabilità che l'utente i venga mostrato nella posizione j.

Risulta che:

$$CTR_i = \sum_{j=1}^{m} y_{ij}\alpha_{ij} \ \forall i \in N$$

$$CTR_i \leq \sum_{j=1}^{m} \alpha_{ij} \ \forall i \in N$$

Una descrizione su come poter calcolare queste quantità è presentata in [13].

Regola di pagamento

La regola di pagamento per questo meccanismo consiste nel richiedere, ad ogni inserzionista visualizzato, il pagamento dell'offerta dell'inserzionista che si trova appena sotto di lui nel ranking, più un piccolo incremento (tipicamente 0.01$). Il prezzo per click risulta quindi:

$$p_i(b) = \begin{cases} \sum_j = 1^m \left(b^{(j+1)} y_{ij}(b)\right) & : \text{se } m < n \text{ o } n \leq m \text{ ma } b_i \neq b^{(n)} \\ 0 & : \text{altrimenti} \end{cases}$$

dove $b^{(j+1)}$ è la $(j+1)$-esima offerta più alta, ovvero l'offerta dell'inserzionista mostrato in posizione $(j+1)$. Per semplicità il piccolo incremento che viene aggiunto è stato omesso dalla formulazione.

2.2.3 Vickrey-Clarke-Groves mechanism

Il meccanismo GSP può apparire una versione generalizzata della famosa asta di Vickrey, utilizzata per la vendita di un singolo bene. In realtà, è stato dimostrato che se i beni da allocare sono più di uno questa uguaglianza non sussiste. La generalizzazione del meccanismo d'asta di Vickrey è invece il meccanismo di Clarke, che andiamo ora a descrivere. È prassi comune riferirsi a questo meccanismo con il nome di Vickrey-Clarke-Groves (VCG) mechanism.

Regola di allocazione

Per definizione, il meccanismo VCG è efficiente allocativamente. La sua regola di allocazione è la seguente:

$$y^*(.) = \arg\max_{y(.)} \sum_{i=1}^{n} b_i v_i(y(b)) = \arg\max_{y(.)} \sum_{i=1}^{n} \sum_{j=1}^{m} (b_i \alpha_{ij}) y_{ij}(b).$$

Tuttavia, sotto l'assunzione che le probabilità di click siano indipendenti dall'identità degli inserzionisti, questa regola si riduce all'allocazione in ordine decrescente di offerta, ovvero dato $b = (b_1, ..., b_n)$:

$$y^*_{ij} = \begin{cases} 1 & : b_i = b^{(j)} \\ 0 & : \text{altrimenti.} \end{cases}$$

Regola di pagamento

Come regola di pagamento viene utilizzata la regola di pagamento di Clarke:

$$t_i(b) = \left[\sum_{j \neq i} b_j v_j(y^*(b))\right] - \left[\sum_{j \neq i} b_j v_j(y^*_{-i}(b))\right]$$

dove $y^*_{-i}(.)$ è un'allocazione di slot efficiente quando l'inserzionista i viene rimosso dal sistema.

2.2.4 GSP vs VCG

Nel caso di un unico bene da allocare, che nel nostro esempio consisterebbe in un unico slot pubblicitario all'interno di una pagina web, i meccanismi GSP e VCG sono identici, ovvero le regole di allocazione e pagamento portano allo stesso risultato. Nel caso di vendita multi-bene, invece, il meccanismo GSP rappresenta una generalizzazione del meccanismo VCG, ed essi presentano delle differenze significative:

- Il meccanismo VCG è incentive compatible in strategie dominanti (DSIC), ovvero dire la verità rappresenta per gli agenti la strategia dominante ed è quindi un punto di equilibrio. Questo non vale per il GSP, che non è nemmeno Bayesian incentive compatible.
- Nel meccanismo GSP l'inserzionista in posizione j paga esattamente quanto offerto da quello in posizione $j + 1$. Per VCG questo prezzo è leggermente aumentato, in quanto nel calcolo del pagamento si considera anche che occupando una certa posizione questa non sia più libera per altri inserzionisti.

Attualmente, i principali sistemi commerciali che implementano una sponsored search auction utilizzano GSP. Il meccanismo VCG è molto studiato, ma per diversi motivi ad oggi ancora non utilizzato.

2.3 Ricerca tramite integratori

Questo lavoro di tesi si pone all'interno di un nuovo paradigma per la composizione di servizi di ricerca che prende il nome di Search Computing (SECO) [17]. Lo scopo di questo progetto è la creazione di un servizio che si collochi a metà strada tra i motori di ricerca *general purpose*, che spesso possono risultare imprecisi, e quelli dedicati ad uno specifico dominio, precisi ma limitati dal dominio stesso, presentandone i pregi di entrambi. Questo servizio si basa sul guidare l'utente nella composizione e nell'utilizzo di query multi-dominio che si appoggiano a diversi servizi di ricerca sottostanti. Tipici esempi di query che l'utente può comporre sono "Dove si trova il cinema più vicino al mio hotel, che offre un film con un giudizio elevato e che si trova vicino ad una pizzeria?", "Chi è il miglior dottore che può curare l'insonnia in un ospedale pubblico nelle vicinanze?", "Quali sono i più elevati fattori di rischio associati alle più diffuse malattie tra la popolazione

giovane?". Il problema principale nell'utilizzo di query multiple risiede nell'estrazione di informazioni da dati complessi anzichè da una singola pagina Web. Questi dati, infatti, richiedono un adeguato processo di integrazione. SECO provvede quindi a fornire una serie di servizi volti ad orchestrare vari motori di ricerca general-purpose e domain-specific per costruire un risultato globale che viene presentato all'utente. Descriveremo ora brevemente due servizi che appartengono al framework di SECO, ovvero l'utilizzo dei service marts e il paradigma Liquid Query.

2.3.1 Service Marts

Abbiamo appena descritto come il concetto fondamentale del paradigma Search Computing sia l'integrazione di diversi servizi di ricerca per migliorare la qualità dei risultati forniti all'utente. Per poter integrare questi servizi di ricerca, risulta necessario che essi vengano descritti univocamente tramite un'interfaccia, nascondendo quindi la loro implementazione. Per service mart [1] si intende quindi un semplice schema tramite il quale ogni oggetto Web può essere mappato. Associare un oggetto a un service mart consiste nel nasconderne la struttura dati sottostante presentando un'interfaccia costituita da attributi di input, output e di ordinamento. Vediamo ora una definizione più precisa di service marts, tramite i tre livelli a cui essi possono essere descritti.

Livello concettuale

Ogni service mart è un'astrazione che descrive una classe di oggetti Web, e viene definito da un nome e una serie di attributi, atomici o multi-valore. Alcuni esempi di schema sono:

$Movie(Title, Director, Score, Year, Genres(Genre), Language,$
$Openings(Country, Date), Actors(Name))$
$Cinema(Name, Address, City, Country, Phone, Movies(Title, StartTimes))$
$Restaurant(Name, Address, City, Country, Phone, Url, Rating, Category(Name))$

La definizione dello schema è semplice e intuitiva. Notiamo che tramite gli attributi multi-valore è possibile modellare relazioni $1{:}M$ o $M{:}N$, in modo da collegare logicamente tra loro oggetti del mondo reale (ad esempio la relazione "recita-in" tra "actor" e "movie").

Livello logico

A livello logico ogni service mart è associato a uno o più *access patterns* specifici. Essi specificano il modo in cui è possibile accedere al service mart, definendo gli attributi di un oggetto come di input (I) oppure di output (O). Un attributo di output può a sua volta essere definito come ranked (R) se il servizio produce il risultato in un ordine che dipende da quell'attributo. Vediamo due access pattern di esempio per l'oggetto "Movie":

$Movie_1(Title^O, Director^O, Score^R, Year^O, Genres(Genre)^I, Language^O,$
$Openings.Country^I, Openings.Date^I, Actor.Name^O)$
$Movie_2(Title^I, Director^O, Score^R, Year^O, Genres(Genre)^O, Language^O,$
$Openings.Country^I, Openings.Date^I, Actor.Name^O)$

Il primo access pattern implementa la ricerca "film recenti per genere", in quanto richiede in ingresso il genere del film oltre che la data e il paese. Nel secondo access pattern invece in input viene richiesto il nome del film, implementando così la ricerca "film recenti per titolo". L'attributo di output *"Score"* viene in entrambi i casi utilizzato per ordinare i film risultanti in ordine decrescente. Modificando le proprietà degli attributi da input a output o viceversa possiamo definire a nostro piacimento l'accesso ai dati che lo specifico servizio richiede.

Connection patterns

I connection patterns introducono un accoppiamento tra due service marts in relazione tra di loro. Vediamo un esempio:

$Shows(Movie, Cinema) : [(Title = Title)]$

Si vede subito come essi rappresentino il concetto di *join* proprio delle basi di dati: se il titolo del film è uguale ad uno dei titoli dei film mostrati nel cinema, allora il predicato è soddisfatto e le istanze degli oggetti *"Movie"* e *"Cinema"* sono composte per formare il risultato. I connection patterns possono essere *directed* se uno degli operandi è un attributo di input, oppure *undirected* se sono entrambi di output (come nell'esempio). Non possono esistere predicati basati su due attributi di input.

Livello fisico

A livello fisico vengono modellate le cosiddette *service interfaces*, create in modo da poter essere mappate su fonti di dati concrete. Esse vengono definite da quattro parametri:

- **Ranking descriptors** Classificano l'interfaccia di servizio come un *search service*, se produce risultati ordinati, oppure un *exact service*, se produce risultati non ordinati.

- **Chunk descriptors** Descrivono come l'interfaccia di servizio produca l'output. Il servizio è *chunked* se può venire invocato ripetutamente e ad ogni invocazione produce un nuovo insieme finito di oggetti. Il numero di tuple ritornate è chiamato *chunk size*. Una differenza fondamentale tra un search service e un exact service è che nel secondo tutti i *chunks* ritornati devono essere considerati, mentre nel primo ci si può limitare ad esaminare solo i primi, sapendo che contengono i risultati più significativi.

- **Cache descriptors** Servono ad ottimizzare il servizio nel caso di invocazioni ripetute su di esso, memorizzando lato-client le risposte fornite da alcuni input, che possono essere utilizzate per successive ricerche. Non sempre l'interfaccia risulta essere *cacheable*, ad esempio nel caso di servizi real-time.

- **Cost descriptors** Descrivono il costo di ogni chiamata ad uno specifico servizio, in termini di tempo di risposta oppure di costo monetario nel caso di servizi a pagamento.

Abbiamo visto come tramite i service marts sia possibile definire l'interfaccia di diversi servizi per garantire interoperabilità tra di essi. Vediamo ora come SECO prevede di aiutare e guidare l'utente nella composizione di query multi-dominio, tramite il paradigma Liquid Query.

2.3.2 Liquid query

Abbiamo già accennato alla necessità di effettuare query multi-dominio. La ricerca su motori dedicati ad un particolare dominio produce risultati più precisi e dettagliati, ed è quindi preferibile. Tuttavia, questi motori risultano essere limitati dal dominio stesso. Si può pensare che un utente esperto possa eseguire più ricerche utilizzando diversi motori specifici e combinando i risultati manualmente, ma la qualità dei risultati prodotti non si avvicinerebbe

a quella ottenuta tramite un efficente *join* automatico dei dati provenienti dai diversi processi di ricerca.

Per questo motivo il paradigma Liquid Query [4] è stato progettato per permettere agli utenti di formulare query multi-dominio tramite un approccio basato sulla nozione di *exploratory search*, che descrive la situazione in cui un utente scopre progressivamente, tramite l'interazione con il servizio, ciò di cui ha bisogno e le informazioni necessarie a soddisfare il bisogno stesso.

Descriviamo brevemente il ciclo di vita di una tipica query:

- **Application configuration phase** Un utente esperto crea un *liquid query template*, specificando i servizi di ricerca che forniranno i risultati e la definizione dell'interfaccia utente.
- **Query submission phase** L'utente invoca una *liquid query* iniziale, basata su un certo template creato in precedenza.
- **Query execution phase** La query viene eseguita e il risultato prodotto viene presentato all'utente tramite l'interfaccia definita.
- **Result browsing phase** Il servizio non si limita a presentare i dati ritornati dalla ricerca. Tramite un insieme di primitive di interazione, infatti, permette all'utente di modificare le caratteristiche della ricerca, i risultati forniti o il modo in cui essi vengono presentati. Le primitive di interazione possono essere di tipo server-side (Remote Interaction Primitive) oppure possono riguardare solo la riorganizzazione dei dati client-side (Local Interaction Primitives).

Abbiamo fornito una breve descrizione di due dei principali servizi che compongono il framework di SECO per presentare il concetto principale su cui esso si basa, ovvero l'integrazione tra servizi di ricerca eterogenei al fine di migliorare la qualità e la precisione dei risultati forniti all'utente. Nel prossimo capitolo descriveremo la nostra proposta per affiancare a questo servizio un sistema commerciale di sponsored search, basato sull'utilizzo dei link pubblicitari risultanti dai motori di ricerca utilizzati, che possa garantire profitti a questo innovativo paradigma.

Capitolo 3

Integrazione di link sponsorizzati

Vogliamo introdurre l'idea fondamentale che sta alla base della formulazione del modello economico proposto. Per motivare questa scelta, presentiamo prima un breve excursus sulle soluzioni commerciali proposte fino ad oggi nel campo dei motori di ricerca. Descriviamo poi come questo processo di sviluppo abbia portato a formulare la nostra proposta di integrazione. Infine, presentiamo le caratteristiche principali che andremo a richiedere, e mostriamo come la descrizione tramite un meccanismo di automated mechanism design, introdotto nel precedente capitolo, rappresenti una formulazione naturale per lo scenario considerato.

3.1 Utilizzo commerciale dei motori di ricerca

L'industria dei motori di ricerca ha avuto inizio nel 1990 con il rilascio di Archie, il primo strumento per ricercare contenuti su Internet. Da quel momento in poi, la richiesta di mercato di tali strumenti è cresciuta continuamente, ed al giorno d'oggi le compagnie che offrono soluzioni di questo tipo sono sempre più diffuse. Per avere un'idea, nei soli Stati Uniti vengono effettuate circa 13 miliardi di ricerche al mese. Tuttavia, come per molte delle società che operano su Internet, il problema principale di queste compagnie consisteva e consiste tuttora nel riuscire a monetizzare sufficientemente il loro grande valore. Una delle soluzioni più significative a questo problema è rappresentata dal concetto di ricerca sponsorizzata, che abbiamo già introdotto nel Capitolo 2.

3.1.1 Storia

Inizialmente, i meccanismi pubblicitari su Internet si ispiravano a quelli del mondo reale. I famosi banner pubblicitari furono introdotti da HotWired nel 1994 [6], ed inizialmente venivano pagati solo per numero di visualizzazioni. Prima dell'introduzione della ricerca sponsorizzata, anche i motori di ricerca utilizzavano questi banner per generare profitti. Essi comportavano tuttavia un dilemma: il motore di ricerca era combattuto tra indirizzare l'utente verso i siti risultanti dalla ricerca o cercare in qualche modo di trattenerlo al fine di mostrargli più inserzioni sponsorizzate possibili. Questo problema portò alla nascita del concetto di *paid search*, che legava l'indirizzamento verso un certo sito al pagamento monetario di una certa quantità al motore di ricerca. Nel 1996, il motore di ricerca Open Text introdusse il concetto di *preferred listings*, tramite i quali si forniva la possibilità ad un inserzionista di pagare per essere inserito nei risultati di ricerca di una particolare parola. Questa innovazione fece sostanzialmente nascere il concetto di ricerca sponsorizzata che oggi conosciamo. Da quel momento, si è assistito allo sviluppo di sistemi sempre più sofisticati, al fine di generare maggiori profitti. Compito del motore di ricerca è quello di decidere come determinare quanto ogni inserzionista debba pagare. Diverse soluzioni sono state proposte per questo calcolo, tra le principali ricordiamo:

- *cost per mille (CPM)* costo per la visualizzazione per mille volte di un link sponsorizzato.
- *cost per click (CPC)* costo per singolo click.
- *cost per action* (CPA) costo per azione, che può essere la registrazione di un nuovo utente o l'acquisto di un bene sul sito dell'inserzionista.

Il passo successivo è stata l'introduzione dell'ordinamento dei risultati visualizzati, non solo in base alla corrispondenza del contenuto, ma anche all'offerta proposta dall'inserzionista. Questa necessità di ordinamento ha portato alla creazione di meccanismi d'asta, utilizzati per primi da FlyCast e Narrowline nel 1997 [19], per il calcolo del posizionamento dei link sponsorizzati. Un nuovo valore chiamato *click through rate (CTR)*, che abbiamo già definito nel precedente capitolo, è stato introdotto e utilizzato per calcolare gli inserzionisti vincenti, nel 2000 da Direct Hit [6] e nel 2002 dal nuovo Google Adwords [8].

L'attuale situazione commerciale dei motori di ricerca vede senza dubbio una posizione dominante di Google, che detiene circa l'84% del totale di mercato. Yahoo! si assesta sul 6% circa, mentre gli altri motori non superano

il 3%. Forniremo ora la descrizione di una delle soluzioni commerciali più significative che Google propone, ovvero Google Adsense.

3.1.2 Un esempio: Google Adsense

Google AdSense [7] rappresenta uno dei più importanti strumenti pubblicitari creati dal colosso di Mountain View. Descriveremo brevemente come funziona, per introdurre un'idea che svilupperemo più avanti nel corso della trattazione, ovvero la scelta di visualizzare sul proprio sito link sponsorizzati registrati ad un altro sito, in questo caso un motore di ricerca, per ricevere un guadagno. Il funzionamento di AdSense, come per la maggior parte dei sistemi pubblicitari, è piuttosto intuitivo. Un webmaster che voglia utilizzare questo servizio si registra a Google e inserisce all'interno del proprio sito una parte di codice Javascript che identifica un annuncio pubblicitario, configurandone il layout a proprio piacimento. A questo punto, AdSense utilizza un sistema di analisi personalizzato, che permette di estrarre dai contenuti del sito registrato alcune parole chiave, che rappresentano gli argomenti principali di cui il sito tratta. Queste parole chiave, tramite un altro servizio di Google chiamato AdWords [8], vengono utilizzate per generare contenuti pubblicitari attinenti al sito e visualizzarli sulla pagina del richiedente. La scelta degli annunci da pubblicare viene fatta principalmente in base a due criteri: il targeting contestuale, ovvero l'analisi delle parole chiave, la frequenza dei termini, la dimensione del carattere e la struttura complessiva dei collegamenti; il targeting per posizionamento, che permette agli inserzionisti di scegliere specifici posizionamenti di annunci su particolari siti o sottosezioni di siti appartenenti alla rete Adsense. Il sistema di analisi individua anche la lingua principale del sito e, se si tratta di una lingua supportata, pubblica annunci in quella lingua.

A questo punto il sistema pubblicitario è pronto per essere utilizzato. Se un visitatore del sito clicca sull'annuncio di un inserzionista, questi pagherà Google secondo quanto stabilito nel contratto, ed una percentuale di questo pagamento andrà al proprietario del sito. Non è possibile stabilire esattamente a quanto ammonti questa percentuale perchè essa viene calcolata tramite algoritmi piuttosto complessi, e dipende tra le altre cose dalla modifica del budget degli inserzionisti, dalla natura dei contenuti e dal tipo di annuncio cliccato. Oltre a questo servizio, che prende il nome di AdSense per i contenuti, Google offre anche ai webmaster interessati la possibilità di inserire all'interno del proprio sito una casella di ricerca, che produrrà link sponsorizzati allo stesso modo di come avviene utilizzando la pagina

principale del motore. Questo servizio prende il nome di AdSense per la ricerca.

Come detto nell'introduzione di questo esempio, abbiamo trattato questa soluzione commerciale perchè introduce la seguente idea innovativa: per poter creare un sistema pubblicitario per una piattaforma web, non abbiamo necessariamente bisogno di ricercare direttamente degli inserzionisti che vogliano pubblicare annunci sul nostro sito. Possiamo invece mettere a disposizione il nostro spazio ad altri sistemi commerciali, che sono tenuti a pagarci una percentuale del guadagno prodotto se uno degli annunci pubblicati viene cliccato. Vedremo come il modello che andremo a proporre si basi proprio su questo concetto.

3.1.3 Definizione

Prima di definire la nostra proposta di integrazione, riassumiamo gli elementi principali della ricerca sponsorizzata:

- Gli inserzionisti forniscono i contenuti, ovvero una serie di annunci caratterizzati da parole chiave, titolo e descrizione.
- Gli inserzionisti specificano delle offerte per ogni parola chiave.
- Un processo di revisione, che può essere automatico o manuale, verifica che il contenuto delle inserzioni sia effettivamente rilevante per le parole chiave.
- Ad ogni ricerca effettuata, il motore seleziona il contenuto pubblicitario relativo alla particolare parola chiave.
- Le inserzioni vengono visualizzate secondo un certo ordine, in una zona della pagina separata oppure combinate assieme a link non sponsorizzati.
- I dati statistici che vengono continuamente rilevati sono utilizzati per calcolare quanto ogni inserzionista debba pagare.

3.2 Una proposta di integrazione

Abbiamo finora parlato delle principali caratteristiche dei sistemi pubblicitari per motori di ricerca. Nell'ultima sezione del Capitolo 2, abbiamo invece introdotto il paradigma Search Computing, che presenta un'innovativa proposta di integrazione tra motori di ricerca aventi dominio specifico, al fine di migliorare i risultati forniti all'utente. È evidente che questo meta-motore

necessiterà a sua volta di un sistema pubblicitario per generare profitti. Ci chiediamo quindi quale sistema commerciale sia possibile proporre per questo integratore. Un'alternativa possibile sarebbe quella di implementare un classico meccanismo di ricerca sponsorizzata. Ogni inserzionista, a cui interessi comparire nella lista dei link sponsorizzati di SeCo, avrebbe quindi la possibilità di registrarsi e specificare la propria offerta per le particolari parole a cui è interessato. Ogni volta che una parola viene ricercata, un meccanismo d'asta stabilirebbe la lista degli inserzionisti vincenti, che verrebbero mostrati a fianco dei risultati dell'integratore.

Sebbene questa possa essere una scelta percorribile, questo lavoro di tesi si concentra invece su una soluzione commerciale differente. Consideriamo infatti i motori di ricerca utilizzati da SeCo. Ad ogni ricerca ognuno di essi produce una lista di link sponsorizzati, prodotta dal particolare meccanismo d'asta implementato. L'idea del modello che descriveremo consiste nel combinare queste liste producendo, allo stesso modo di ciò che viene fatto con i risultati, un'integrazione tra i diversi motori, al fine di ottenere un'unica lista di link sponsorizzati. In breve, il meccanismo proposto funziona nel seguente modo:

- Nel corso di una ricerca, l'integratore interroga i diversi motori che, oltre a fornire i risultati veri e propri, gli comunicano di volta in volta la propria lista di link sponsorizzati, insieme ad alcuni valori che caratterizzano gli inserzionisti visualizzati.

- L'integratore combina secondo dei propri criteri le liste di link sponsorizzati che gli sono state comunicate da ogni singolo motore, producendo e visualizzando la propria lista.

- Se un link visualizzato dall'integratore viene cliccato, l'inserzionista pagherà al motore di ricerca a cui esso è registrato un certo valore. Il motore di ricerca in questione ritornerà una percentuale di questo pagamento verso l'integratore, che a sua volta provvederà a ridistribuire questo pagamento tra i diversi motori di ricerca che prendono parte al meccanismo.

I punti critici della trattazione, che andremo ora ad approfondire, sono la comunicazione delle caratteristiche degli inserzionisti, i criteri secondo cui la lista finale viene prodotta e il calcolo dei pagamenti.

Per concludere, precisiamo che abbiamo descritto il modello proposto come alternativa ad un classico meccanismo in cui gli inserzionisti si registrano direttamente all'integratore, ma tuttavia queste due soluzioni non si escludono. Sviluppi futuri potrebbero portare ad un sistema di integrazione

capace di combinare le liste prodotte da ogni motore di ricerca consultato con le offerte di un insieme di inserzionisti registrati direttamente all'integratore.

3.2.1 Problema delle vere informazioni

Nella sezione precedente abbiamo fornito un'idea generale di come il meccanismo, che descriveremo formalmente nel Capitolo 4, funzioni. Ci concentriamo ora sulle informazioni di cui l'integratore necessita per potere combinare le liste di link sponsorizzati che gli vengono comunicate, al fine di generare la propria. Abbiamo visto come i meccanismi d'asta dei principali motori di ricerca presenti in letteratura si basino sostanzialmente su due variabili, ovvero il valore dell'offerta e la probabilità di click. Precisiamo che la prima quantità è esattamente determinata dal contratto stipulato tra inserzionista e motore di ricerca, mentre la seconda viene stimata dal motore stesso in base a dati statistici. Per poter rendere possibile l'implementazione del meccanismo descritto, è necessario che l'integratore venga a conoscenza di questi valori, per ogni inserzionista che potrebbe essere visualizzato. Ogni motore di ricerca utilizzato, quindi, oltre a fornire la propria lista, deve comunicare i valori di offerta e probabilità di click per ogni inserzionista presente in essa. Questi dati sono informazioni private, conosciute con esattezza solo dal singolo motore di ricerca. È possibile garantire in base ad un contratto che ogni motore comunichi i valori relativi ad ogni inserzionista, ma cosa garantisce che questi valori siano veramente quelli che possiede? Notiamo che ci siamo ricondotti ad un gioco in forma strategica, dove gli agenti corrispondono ai diversi motori di ricerca, le azioni alle comunicazioni dei valori, e gli esiti all'assegnazione delle posizioni nella lista dei link sponsorizzati. Come tutti i giochi strategici, anche questo viene definito da regole, che è compito dell'integratore creare. Queste regole dovranno essere formulate in maniera tale da garantire che, in ogni situazione, la strategia migliore per ogni agente in gioco sia quella di comunicare la verità, ovvero gli effettivi valori di offerte e probabilità di click che esso possiede. Vedremo nel Capitolo 4 come vengono inseriti nel modello formale i vincoli che garantiscono questa proprietà.

3.2.2 Formulazione AMD

Abbiamo visto come il modello commerciale che vogliamo andare a proporre si possa ricondurre alla risoluzione di un gioco in forma strategica. Per quanto detto, risulta naturale descrivere il problema utilizzando la teoria di mechanism design: ci poniamo infatti nel punto di vista di un social planner, ovvero l'integratore stesso, che riceve le preferenze degli agenti in gioco sottoforma di valori comunicati e affronta il problema di prendere una

decisione, sapendo che le informazioni ricevute potrebbero non corrispondere a verità. Inoltre, trattandosi di un sistema commerciale, ciò che si vuole ottenere non risulta essere il raggiungimento di un punto di equilibrio, bensì la massimizzazione di una certa funzione obiettivo, ad esempio il profitto dell'integratore oppure il profitto complessivo di integratore e motori di ricerca. Questi motivi giustificano la scelta della formulazione del modello come un problema di *automated mechanism design*, che abbiamo descritto nel precedente capitolo. Un ulteriore vantaggio che l'utilizzo di questo meccanismo comporta è la flessibilità che esso garantisce nel corso del processo di creazione. Lo studio di un nuovo modello, infatti, necessita sempre di continui cambiamenti ed adattamenti in corso d'opera, man mano che la sua formulazione si avvicina sempre più alla soluzione migliore. Il grande vantaggio della formulazione AMD è quello di permettere la modifica di variabili in gioco, funzioni obiettivo e parametri, senza la necessità di riformulare ogni volta da zero un diverso modello.

3.2.3 Meccanismo

Prima di concludere con una breve descrizione del meccanismo implementato, precisiamo come avviene la stima della probabilità di click per ogni inserzionista. Il calcolo di questo valore non è banale, perchè anche da esso dipende l'esito dell'asta. Sappiamo che l'integratore ha a disposizione le probabilità di click comunicategli da ogni motore di ricerca, e deve in qualche modo combinare questi valori per ottenere una stima complessiva. Nella nostra trattazione utilizzeremo un valore semplice, ovvero la media delle probabilità di click calcolata su tutti i motori di ricerca. Sviluppi futuri potrebbero portare ad implementare una diversa funzione di calcolo, che possa migliorare la funzione obiettivo che vogliamo massimizzare. Una prima modifica potrebbe consistere nell'assegnare diversi pesi a diversi motori di ricerca, a seconda dell'importanza del motore o di altri criteri. Una seconda modifica potrebbe considerare, nel corso del calcolo, unicamente i motori di ricerca che l'utente sta effettivamente utilizzando. Poichè, infatti, il paradigma SeCo prevede di guidare l'utente nella ricerca di migliori risultati, nel corso dell'interazione potrebbe rendersi necessario aggiungere o rimuovere alcuni dei motori utilizzati. Ognuna di queste azioni potrebbe produrre un ricalcolo delle probabilità di click e la conseguente visualizzazione di una nuova lista di link sponsorizzati.

Abbiamo ora definito tutti gli elementi in gioco. Riassumiamo il modello commerciale proposto:

- L'utente esegue una ricerca su SeCo, che utilizza una serie di motori di ricerca domain-specific per comporre i risultati.
- Ogni motore di ricerca utilizzato comunica all'integratore le proprie informazioni private (ovvero offerte e probabilità di click) relative alla lista di link sponsorizzati che la sua ricerca ha prodotto. Il meccanismo di incentivi realizzato dall'integratore deve garantire che vengano comunicate le vere informazioni.
- L'integratore calcola, in base ai valori delle probabilità di click ricevuti, una stima della probabilità di click per ogni inserzionista.
- L'integratore produce una lista di link sponsorizzati, scelti in base alla massimizzazione di una certa funzione obiettivo e nel rispetto dei vincoli imposti dal problema.
- L'integratore interagisce con l'utente nel corso della ricerca, aggiornando i valori delle probabilità di click e quindi la lista dei link sponsorizzati ogni volta che un motore viene aggiunto o rimosso.

Una rappresentazione del modello descritto è rappresentata in Figura 3.1

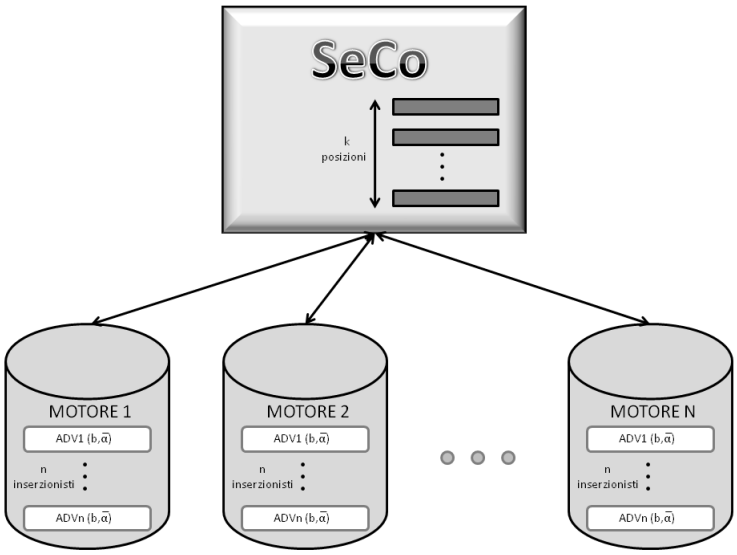

Figura 3.1: Una rappresentazione del modello dell'integratore

Capitolo 4

Analisi teorica

Nel precedente capitolo abbiamo descritto la proposta di integrazione che vogliamo realizzare. Abbiamo inoltre individuato una classe di meccanismi a cui lo scenario considerato si adatta naturalmente, ovvero i meccanismi AMD. Il nostro modello è pronto per essere formalizzato. Dopo aver definito gli agenti in gioco e come essi possano interagire tra di loro, ci concentriamo sulle proprietà che vogliamo ottenere dalla definizione delle regole del gioco. Introduciamo il concetto di valutazioni interdipendenti, e vediamo come, in questa situazione, alcune delle proprietà richieste risultino essere in conflitto tra di loro, ovvero non simultaneamente soddisfacibili nel caso generale. Queste considerazioni rappresentano un punto di partenza per sviluppi futuri del meccanismo descritto.

4.1 Il meccanismo formale

Introduciamo le variabili di cui avremo bisogno per formalizzare il meccanismo. Definiamo innanzitutto l'insieme S dei motori di ricerca *domain-specific* che SeCo si propone di integrare. Ad ogni motore di ricerca è registrato un certo numero di inserzionisti. Chiamiamo A l'insieme complessivo di tutti gli inserzionisti e $A(s)$ con $s \in S$ quelli che appaiono nella lista dei link sponsorizzati di un singolo motore s. Ogni inserzionista è caratterizzato da una determinata offerta b e da una probabilità di click α, che sono informazioni private del motore di ricerca a cui essi sono registrati. Abbiamo introdotto nel Capitolo 2 il concetto di *tipo privato* di un agente. Modelliamo quindi il comportamento di ogni inserzionista riguardo alla particolare parola ricercata definendo una variabile $\theta_{s,a}$, che rappresenta il tipo privato di a registrato al motore s. L'insieme di tutti i possibili tipi è $\Theta_{s,a}$. Ad ogni tipo

$\theta_{s,a} \in \Theta_{s,a}$ è associata una certa offerta b ed una certa probabilità di click α. Definiamo quindi le funzioni $b(\theta_{s,a}) : \Theta_{s,a} \to \mathbb{R}$ e $\alpha(\theta_{s,a}) : \Theta_{s,a} \to [0, 1]$ che ritornano rispettivamente l'offerta e la probabilità di click dell'inserzionista a registrato al motore s quando il suo tipo privato è $\theta_{s,a}$. Ogni inserzionista a può comparire in più di un motore di ricerca con valori anche differenti di offerta e probabilità di click, ovvero $b(\theta_{s,a})$ può essere diverso da $b(\theta_{s',a})$ così come $\alpha(\theta_{s,a})$ può essere diverso da $\alpha(\theta_{s',a})$. Per completare la descrizione di ogni singolo inserzionista, definiamo con $\omega(\theta_{s,a})$ la probabilità che esso abbia tipo privato $\theta_{s,a}$.

È utile chiarire che, nella nostra trattazione, le informazioni di offerta e di probabilità di click di ogni inserzionista si suppongono essere già determinate (ma sconosciute a SECO) dal meccanismo che ogni motore di ricerca adotta per visualizzare la propria lista di link sponsorizzati. Non ci preoccuperemo infatti di incentivare ogni singolo inserzionista a non mentire, perchè questo viene già garantito dai meccanismi d'asta che il motore di ricerca a cui esso è registrato utilizza. Quello che vogliamo ottenere è la comunicazione veritiera delle informazioni di offerta e probabilità di click da parte del motore di ricerca, per ogni inserzionista che esso ha deciso di visualizzare. Abbiamo visto come queste informazioni siano riassunte nel concetto di tipo $\theta_{s,a}$. Possiamo quindi definire ogni motore di ricerca come un agente in gioco, avente come tipo privato $\theta_s = (\theta_{s,1}, \ldots, \theta_{s,|A(s)|})$, ovvero il vettore di tipi degli inserzionisti visualizzati tra i propri link sponsorizzati. L'insieme di tutti i possibili tipi del motore s è Θ_s. Allo stesso modo $\omega(\theta_s) = \prod_{\theta_{s,a} \in \theta_s} \omega(\theta_{s,a})$ rappresenta la probabilità che il motore di ricerca s presenti tipo θ_s. Abbiamo così definito una distribuzione di probabilità su Θ_s, ovvero sul profilo di ogni agente in gioco, come prodotto di un insieme di distribuzioni di probabilità indipendenti. Indichiamo il profilo di tutti gli agenti con $\theta \in \Theta$, dove l'insieme Θ è composto da tutti gli insiemi Θ_s. Riassumiamo le variabili finora definite:

- Ogni inserzionista $a \in A(s)$ è caratterizzato da un tipo $\theta_{s,a}$, che descrive un'offerta $b(\theta_{s,a})$ e una probabilità di click $\alpha(\theta_{s,a})$, e si presenta con probabilità $\omega(\theta_{s,a})$.

- Ogni motore di ricerca $s \in S$ è caratterizzato da un tipo θ_s, che si presenta con probabilità $\omega(\theta_s)$ e descrive l'insieme di tipi degli inserzionisti che esso visualizza.

- Il meccanismo dell'integratore lavorerà sul profilo $\theta \in \Theta$ ricevuto, che rappresenta l'insieme di tipi dei motori di ricerca che andrà a combinare, ovvero $\theta = (\theta_{s_1}, \ldots, \theta_{s_{|S|}})$.

Abbiamo completato la definizione degli agenti in gioco. Specifichiamo ora un esito $x \in X$, che rappresenta l'insieme dei vincitori che verranno mostrati dall'integratore nelle posizioni disponibili. Il numero di posizioni a disposizione è fissato e pari a k. Definiamo quindi formalmente un esito come $x = \langle (s, a), \ldots, (s', a') \rangle$, dove il primo elemento rappresenta il vincitore visualizzato in prima posizione, il secondo in seconda posizione e così via. Nello specificare ogni vincitore, è necessaria la coppia inserzionista-motore, perchè abbiamo visto che un inserzionista può essere registrato a più motori di ricerca con diverse offerte e probabilità di click. Un vincolo che dobbiamo imporre è che lo stesso inserzionista non venga visualizzato in più posizioni, ovvero che per ogni a e a' di uno stesso esito x si abbia $a \neq a'$.

Per ogni alternativa $x \in X$ e ogni motore di ricerca s, è necessario calcolare la valutazione $v_s(x) : X \to \mathbb{R}$ che esso assegna a quel particolare esito. Se s non appare in x, $v_s(x) = 0$. Se invece s appare in x, $v_s(x)$ viene calcolata nel seguente modo: per ogni coppia $(s, a) \in x$, la valutazione attesa di s è il prodotto tra la probabilità di click dell'inserzionista a ed il valore che s riceve quando il link corrispondente ad a viene cliccato. Prima di definire formalmente $v_s(x)$, dobbiamo quindi concentrarci su come l'integratore possa calcolare queste due variabili.

Per quanto riguarda la probabilità di click, essa è relativa ad ogni inserzionista ed è una funzione calcolata sulle $\alpha(\theta_{s',a})$ per ogni motore di ricerca s', dove $\theta_{s',a}$ è il tipo riportato. L'integratore dovrà quindi stimare questa probabilità combinando con un determinato criterio le informazioni a sua disposizione, ovvero le probabilità di click dell'inserzionista a, che vengono comunicate dai motori di ricerca a cui esso è registrato. Chiamiamo questa stima $\overline{\alpha}(a)$. Il calcolo di questo valore può tenere conto di diversi parametri, ad esempio è possibile pesare la probabilità di click comunicata, a seconda dell'importanza che l'integratore attribuisce ad ogni motore di ricerca. È inoltre possibile ricalcolare questo valore ogni volta che, durante l'iterazione tra utente ed integratore, un certo motore viene rimosso o aggiunto da quelli utilizzati per la ricerca. Come già detto, nel nostro lavoro utilizzeremo un semplice stimatore, ovvero la media $\overline{\alpha}(a) = \frac{\sum_{s \in S : a \in A(s)} \alpha(\theta_{s,a})}{|A|}$.

Occupiamoci ora del valore che s riceve quando il link corrispondente ad a viene cliccato. Assumeremo nella trattazione che questo valore corrisponda esattamente all'offerta b. Questo approccio, sostanzialmente di tipo *first-price*, è stato utilizzato per semplicità in modo da potersi concentrare, per questo primo modello, sull'interazione tra motori di ricerca ed integratore. Un possibile sviluppo futuro sarebbe quello di considerare anche il meccanismo con cui ogni singolo motore di ricerca ottiene le informazioni dai propri

inserzionisti.

Abbiamo definito le variabili di cui avevamo bisogno, ovvero $\overline{\alpha}(a)$ e la valutazione che s assegna al click sull'inserzionista a, che con queste assunzioni coincide con $b(\theta_{s,a})$. Definiamo quindi $v_s(x)$ come $v_s(x) = \sum_{(s,a)\in x} \overline{\alpha}(a) \cdot b(\theta_{s,a})$. Siccome $v_s(x)$ rappresenta la valutazione dell'esito calcolata da ogni motore di ricerca, le variabili $\theta_{s,a}$ rappresentano i veri tipi degli inserzionisti del motore s, di cui solo esso è a conoscenza. L'insieme delle possibili valutazioni è definito come $V = \{v_s : s \in S\}$.

Per concludere la definizione di un meccanismo AMD diretto è necessario specificare la funzione di scelta sociale f e la funzione p che determina i pagamenti. La funzione di scelta sociale f è una funzione $f : \Theta \to X$ che, dato un profilo composto dai tipi di tutti i motori di ricerca, restituisce un esito x. La funzione $p : \Theta \to \mathbb{R}^{k\cdot|S|}$, con k numero di posizioni disponibili, restituisce il valore del pagamento relativo ad ogni motore di ricerca nel caso che uno dei link visualizzati (anche non suo) venga cliccato. L'utilità di ogni motore di ricerca verrà definita come la differenza fra la sua valutazione e il suo pagamento, ottenendo uno spazio quasi-lineare.

Il meccanismo diretto $\mathcal{M}(X, S, \Theta, V, f, p)$ è ora completamente definito. Elenchiamo ora le proprietà che vogliamo ottenere dalla definizione di f e p.

4.1.1 Proprietà richieste

Prima di procedere alla definizione delle proprietà, è necessaria una precisazione sul meccanismo. Osserviamo che i motori di ricerca, nel caso generale, presentano valutazioni interdipendenti [11, 9]. Le valutazioni di un agente si definiscono interdipendenti se dipendono dal tipo privato di ogni agente in gioco. Nella definizione della funzione di valutazione che abbiamo appena fornito, si riscontra questa proprietà in quanto il valore $\overline{\alpha}(a)$ dipende dal profilo θ, ovvero dai tipi di tutti i motori di ricerca. L'unica situazione per cui questo meccanismo non presenterebbe valutazioni interdipendenti sarebbe quella caratterizzata da insiemi disgiunti di inserzionisti per ogni motore di ricerca, ovvero solo se $A(s) \cap A(s') = \emptyset \; \forall s \neq s'$. Vedremo come questa caratteristica influenzi il vincolo di incentive compatibility e l'efficienza del meccanismo.

Definiamo ora le proprietà richieste:

- **(Ex-post) individual rationality** Come abbiamo già introdotto nel Capitolo 2, vogliamo che gli agenti, ovvero i motori di ricerca, siano incentivati a partecipare al gioco. Per garantire questa proprietà dobbiamo richiedere che, per ogni possibile esito x, l'utilità di ogni motore

sia non-negativa. Questo richiede un vincolo sulla funzione di pagamento, che imponga un limite alla quantità che il motore di ricerca, a cui l'inserzionista cliccato appartiene, deve pagare all'integratore. Bisogna quindi garantire che, se il link cliccato appartiene all'inserzionista a registrato al motore di ricerca s, il pagamento di s non debba essere maggiore dell'offerta $b(\theta_{s,a})$, ovvero di quanto il motore di ricerca guadagnerà dal proprio inserzionista. Per tutti gli altri motori di ricerca, invece, il pagamento p relativo a quel link dev'essere non-positivo, ovvero dev'essere nullo oppure rappresentare un pagamento dall'integratore verso il motore in questione.

- **(Ex-post) Nash o Bayesian-Nash incentive compatibility** Siccome nel meccanismo descritto gli agenti presentano valutazioni interdipendenti, non forniamo per motivi di complessità la formulazione del vincolo di IC in strategie dominanti. Vogliamo invece garantire che dire la verità sia la strategia ottima per ogni agente, assumendo che ogni altro agente dica la verità. Implementeremo quindi le funzioni f e p garantendo le proprietà di *(ex-post)* Nash IC e Bayesian-Nash IC.

- **(Ex-post) Weak budget balance** Affinchè il meccanismo sia vantaggioso per l'integratore, questi non deve mai ricavarne un'utilità negativa. Per questo motivo, per ogni esito x tale che $f(\theta) = x$, la somma complessiva dei pagamenti di tutti i motori di ricerca deve essere non-negativa. Questo impone che, per ogni link cliccato, qualora l'integratore fosse costretto dal meccanismo di incentivi a pagare alcuni motori di ricerca, l'ammontare complessivo di questi pagamenti sia inferiore a quanto esso riceve dal motore di ricerca a cui il link cliccato appartiene. Il valore delle variabili p è positivo se un motore paga una certa quantità all'integratore e negativo se avviene l'opposto. Andremo quindi ad imporre che la somma dei pagamenti dei motori di ricerca, escluso il motore s a cui appartiene l'inserzionista a cliccato, sia maggiore di $-b(\theta_{s,a})$.

- **Ottimalità** La flessibilità della formulazione AMD ci consente di definire più funzioni obiettivo che vorremo di volta in volta massimizzare:

  1. *ex-ante* utilità attesa dell'integratore.
  2. *ex-ante* utilità attesa delle valutazioni complessive dei motori di ricerca.
  3. *ex-ante* utilità attesa complessiva dei motori di ricerca.
  4. *ex-ante* utilità attesa di uno specifico motore di ricerca.

La scelta della funzione obiettivo dipenderà dallo specifico contratto stipulato dall'integratore.

- **Efficienza** Dobbiamo osservare che, nel caso generale, questo meccanismo non può risultare efficiente. Infatti, un meccanismo *one-stage* non può essere incentive compatible ed efficiente se presenta valutazioni interdipendenti e tipi multipli [9]. L'efficienza non è in generale garantita nemmeno per applicazioni molto semplici di questo meccanismo, come ad esempio nel caso in cui ogni agente abbia un unico tipo. Un possibile sviluppo futuro potrebbe essere quello di implementare un meccanismo *two-stage*, per il quale è possibile avere meccanismi incentive compatible ed efficienti, come è dimostrato in [11].

4.1.2 Formulazione in AMD

Forniamo ora la formulazione del meccanismo come un problema di automated mechanism design [15].

Funzione di scelta sociale f

Abbiamo visto nella descrizione del modello che la funzione di scelta sociale f che vogliamo implementare è una funzione $f : \Theta \to X$ che, dato un profilo composto dai tipi di tutti i motori di ricerca, restituisce un esito x. La funzione dipenderà quindi da $\theta = \theta_{s1}, \ldots, \theta_{s|S|}$. Per definire un esito abbiamo bisogno di tre parametri: l'inserzionista a, il motore di ricerca s a cui esso appartiene e la posizione r in cui esso viene mostrato. Formalmente, definiamo f come una collezione di variabili binarie $f_{s,a,\theta,r} \in \{0, 1\}$, dove $f_{s,a,\theta,r} = 1$ se, dato il profilo di tipi θ, l'inserzionista a registrato al motore s viene visualizzato in posizione r. Le posizioni disponibili sono k, per cui $r \in 1, \ldots, k$. Per semplicità, se l'inserzionista a non è registrato al motore di ricerca s, ovvero $a \in A \setminus A(s)$, imponiamo $f_{s,a,\theta,r} = 0$, $b(\theta_{s,a}) = 0$, e $\omega(\theta_{s,a}) = 0$. Risulta necessario definire due vincoli per la funzione di scelta sociale. Abbiamo visto che un inserzionista a può essere registrato a più motori di ricerca, con offerta e probabilità di click diverse. Tuttavia, nella scelta dei vincitori, esso deve poter comparire al più una sola volta. Otteniamo questo vincolo imponendo che:

$$\sum_{r \in R} \sum_{s \in S : a \in A(s)} f_{s,a,\theta,r} \leq 1 \qquad \forall \theta \in \Theta, \forall a \in A \tag{4.1}$$

Il secondo vincolo, molto semplice, impone che in ogni posizione r sia visualizzato esattamente un link sponsorizzato:

$$\sum_{s \in S} \sum_{a \in A} f_{s,a,\theta,r} = 1 \qquad \forall \theta \in \Theta, \forall r \in R \tag{4.2}$$

Funzione di pagamenti p

La funzione $p : \Theta \to \mathbb{R}^{k \cdot |S|}$, restituisce il valore del pagamento relativo ad ogni motore di ricerca, nel momento in cui uno dei link visualizzati (anche non suo) viene cliccato. Una sua definizione potrebbe essere quindi $p_{s,\theta,r} \in \mathbb{R}$, ovvero la quantità pagata (oppure ricevuta se il valore è negativo) dal motore di ricerca s quando il link in posizione r viene cliccato, nel caso il profilo dei tipi sia θ. Per semplificare la formulazione AMD, abbiamo diviso la variabile $p_{s,\theta,r}$ in pagamenti relativi ai singoli inserzionisti del motore di ricerca s, ottenendo le variabili $p_{s,a,\theta,r}$. Il pagamento relativo a un motore di ricerca è calcolato come $p_{s,\theta,r} = \sum_{a \in A(s)} p_{s,a,\theta,r}$ per ogni $s \in S, \theta \in \Theta$, e $r \in R$. La funzione di pagamenti p deve essere limitata dal vincolo di *ex-post* individual rationality definito nella sezione precedente. È necessario quindi imporre che ogni motore di ricerca non possa pagare un click su di un proprio link più del valore che esso assegna a quel click, che nella nostra trattazione corrisponde all'offerta dell'inserzionista cliccato:

$$p_{s,a,\theta,r} \leq b(\theta_{s,a}) \cdot f_{s,a,\theta,r} \qquad \begin{array}{r} \forall s \in S, \forall a \in A(s), \\ \forall \theta \in \Theta, \forall r \in R \end{array} \tag{4.3}$$

Precisiamo che questo vincolo impone anche che il pagamento di s per il click su un inserzionista di un altro motore, ovvero dove $f_{s,a,\theta,r} = 0$, debba essere non-positivo.
Un secondo vincolo impone che un motore di ricerca s non possa ricevere un pagamento quando in posizione r viene visualizzato un proprio link. Se un certo $a \in A(s)$ è stato assegnato alla posizione r, le corrispondenti variabili $p_{s,a,\theta,r}$ devono essere quindi non-negative $\forall a \in A(s)$. Otteniamo:

$$p_{s,a,\theta,r} \geq -M \cdot \left(1 - \sum_{a' \in A(s)} f_{s,a',\theta,r}\right) \qquad \begin{array}{r} \forall s \in S, \forall a \in A(s), \\ \forall \theta \in \Theta, \forall r \in R \end{array} \tag{4.4}$$

dove M è un numero arbitrariamente grande. Come precisato, questo vincolo limita la variabile $p_{s,a,\theta,r}$ solo quando uno degli inserzionisti appartenenti a s è visualizzato in posizione r, ovvero quando $\sum_{a' \in A(s)} f_{s,a',\theta,r} = 1$.

Questi due vincoli descritti hanno quindi il compito di imporre rispettivamente un vincolo superiore e un vincolo inferiore alla funzione che descrive i pagamenti dei diversi motori di ricerca.

Proprietà richieste

Abbiamo già implementato la proprietà di *ex-post* individual rationality. Completiamo la definizione dei vincoli per garantire le proprietà elencate nella Sezione 4.1.1.

(Ex-post) Nash incentive compatibility Prima di fornire la definizione dei vincoli di incentive compatibility, è necessario introdurre una nuova variabile. Abbiamo definito $\overline{\alpha}(a)$ come la probabilità di click calcolata per l'inserzionista a. Introduciamo ora un secondo parametro per questa variabile, definendo $\overline{\alpha}(a, r)$ come la probabilità di click per l'inserzionista a visualizzato in posizione r. Vediamo ora come possiamo descrivere il "dire la verità" o "mentire", per un motore di ricerca s. Supponiamo che il motore in questione sia di tipo θ_s. Per evidenziare questa situazione, possiamo definire il profilo complessivo di tipi come $\theta = (\theta_s, \theta_{-s})$, con θ_{-s} che rappresenta il profilo di tutti gli altri motori di ricerca. Per ogni posizione r, il guadagno del motore di ricerca è definito dalla differenza tra ciò che riceve dai propri inserzionisti e gli eventuali pagamenti, ovvero da:

$$\sum_{a \in A} \left(b(\theta_{s,a}) \cdot f_{s,a,(\theta_s,\theta_{-s}),r} - p_{s,a,(\theta_s,\theta_{-s}),r} \right) \cdot \bar{\alpha}(a) \tag{4.5}$$

Se esso decide di mentire, dichiarando un tipo θ'_s, il guadagno che ne ricava è differente, in quanto le funzioni di scelta sociale e di pagamento cambiano in funzione del profilo di tipi θ, che in questo caso diventerebbe (θ'_s, θ_{-s}). Ciò che non cambia, invece, è il valore dell'offerta $b(\theta_{s,a})$, relativa al vero tipo in cui il motore si trova. Per ogni posizione r, se decide di mentire guadagnerà quindi:

$$\sum_{a \in A} \left(b(\theta_{s,a}) \cdot f_{s,a,(\theta'_s,\theta_{-s}),r} - p_{s,a,(\theta'_s,\theta_{-s}),r} \right) \cdot \bar{\alpha}(a) \tag{4.6}$$

Il vincolo viene quindi definito iterando su tutte le possibili posizioni r e aggiungendo le relative probabilità di click $\overline{\alpha}(a, r)$. Risulta:

$$\begin{aligned} \sum_{r \in R} \sum_{a \in A} \left(b(\theta_{s,a}) \cdot f_{s,a,(\theta_s,\theta_{-s}),r} - p_{s,a,(\theta_s,\theta_{-s}),r} \right) \cdot \bar{\alpha}(a, r) \geq \\ \sum_{r \in R} \sum_{a \in A} \left(b(\theta_{s,a}) \cdot f_{s,a,(\theta'_s,\theta_{-s}),r} - p_{s,a,(\theta'_s,\theta_{-s}),r} \right) \cdot \bar{\alpha}(a, r) \end{aligned} \quad \begin{aligned} &\forall s \in S, \\ &\forall \theta \in \Theta, \\ &\forall \theta'_s \in \Theta_s \end{aligned} \tag{4.7}$$

(Ex-post) Bayesian-Nash incentive compatibility Per ottenere questo vincolo, è necessario probabilizzare su tutti i possibili profili di tipi θ_{-s}. La probabilità che i motori di ricerca escluso s abbiano profilo θ_{-s} è data da $\prod_{s' \in S/\{s\}} \omega(\theta_{s'})$. Abbiamo quindi:

$$\begin{aligned}
&\sum_{\theta_{-s}} \sum_{r \in R} \sum_{a \in A} \Bigg(\Big(b(\theta_{s,a}) \cdot f_{s,a,(\theta_s,\theta_{-s}),r} - \\
&\qquad - p_{s,a,(\theta_s,\theta_{-s}),r} \cdot \bar{\alpha}(a,r) \Big) \cdot \prod_{s' \in S/\{s\}} \omega(\theta_{s'}) \Bigg) \geq \qquad \forall s \in S, \\
&\sum_{\theta_{-s}} \sum_{r \in R} \sum_{a \in A} \Bigg(\Big(b(\theta_{s,a}) \cdot f_{s,a,(\theta'_s,\theta_{-s}),r} - \qquad \forall \theta_s \in \Theta_s, \quad (4.8) \\
&\qquad - p_{s,a,(\theta'_s,\theta_{-s}),r} \Big) \cdot \bar{\alpha}(a,r) \Big) \cdot \prod_{s' \in S/\{s\}} \omega(\theta_{s'}) \Bigg) \qquad \forall \theta'_s \in \Theta_s
\end{aligned}$$

(Ex-post) Weak budget balance Questo vincolo impone semplicemente che, per ogni posizione r e profilo θ, l'ammontare complessivo dei pagamenti non debba essere negativo.

$$\sum_{s \in S} \sum_{a \in A} p_{s,a,\theta,r} \geq 0 \qquad \forall \theta \in \Theta, \forall r \in R \qquad (4.9)$$

Ricordiamo che questa quantità rappresenta l'utilità dell'integratore per la posizione r, dato il profilo θ.

Funzioni obiettivo

Vediamo ora la definizione formale delle quattro funzioni obiettivo elencate in precedenza.

(Ex-ante) Utilità attesa dell'integratore Nella definizione del vincolo di weak budget balance, abbiamo già precisato la quantità che rappresenta l'utilità dell'integratore. Iterando su posizioni e profili, e pesandola con le rispettive probabilità di click e di tipo, otteniamo l'utilità attesa che vogliamo massimizzare:

$$\max \quad \sum_{\theta \in \Theta} \left(\sum_{r \in R} \sum_{s \in S} \sum_{a \in A} p_{s,a,\theta,r} \cdot \overline{\alpha}(a,r) \right) \cdot \prod_{\theta_s \in \theta} \omega(\theta_{s,a}) \qquad (4.10)$$

(Ex-ante) Utilità attesa delle valutazioni complessive dei motori di ricerca La valutazione complessiva di ogni motore di ricerca è data dalla somma delle singole valutazioni dei propri link visualizzati. Iterando su tutti i motori si ottiene:

$$\max \quad \sum_{\theta \in \Theta} \left(\sum_{r \in R} \sum_{s \in S} \sum_{a \in A} b(\theta_{s,a}) \cdot f_{s,a,\theta,r} \cdot \bar{\alpha}(a,r) \right) \cdot \prod_{\theta_s \in \theta} \omega(\theta_{s,a}) \tag{4.11}$$

(Ex-ante) Utilità attesa complessiva dei motori di ricerca L'utilità complessiva attesa di tutti i motori di ricerca è data dalle valutazioni complessive meno i relativi pagamenti, il tutto pesato per le probabilità di click e di tipo. Combiniamo quindi le due precedenti funzioni obiettivo, ottenendo:

$$\max \quad \sum_{\theta \in \Theta} \left(\sum_{r \in R} \sum_{s \in S} \sum_{a \in A} \left(b(\theta_{s,a}) \cdot f_{s,a,\theta,r} - p_{s,a,\theta,r} \right) \cdot \bar{\alpha}(a,r) \right) \cdot \prod_{\theta_s \in \theta} \omega(\theta_{s,a}) \tag{4.12}$$

(Ex-ante) Utilità attesa di uno specifico motore di ricerca Rimuoviamo semplicemente la sommatoria su tutti i motori di ricerca per ottenere l'utilità di un singolo motore s:

$$\max \quad \sum_{\theta \in \Theta} \left(\sum_{r \in R} \sum_{a \in A} \left(b(\theta_{s,a}) \cdot f_{s,a,\theta,r} - p_{s,a,\theta,r} \right) \cdot \bar{\alpha}(a,r) \right) \cdot \prod_{\theta_s \in \theta} \omega(\theta_{s,a}) \tag{4.13}$$

Capitolo 5

Analisi sperimentale

Obiettivo di questo lavoro di tesi è stato principalmente la descrizione formale del modello, descritta nel capitolo precedente. Presentiamo comunque due esempi di applicazione del meccanismo creato, per fornire un'idea di come esso funzioni. Nella seconda parte del capitolo descriviamo invece uno strumento per la generazione automatica di modelli di possibili scenari reali, creato nel corso del processo di sviluppo. Esso fornisce, dati alcuni parametri in ingresso, la formulazione in programmazione matematica del modello e la generazione di dati casuali, che è possibile utilizzare per le sessioni di test necessarie all'analisi del comportamento del meccanismo. Concludiamo il capitolo con i risultati sperimentali relativi ad alcuni semplici scenari creati utilizzando questo generatore.

5.1 Casi di studio notevoli

Presentiamo in questa sezione due semplici applicazioni del modello descritto: esse rappresentano il punto di partenza per lo studio del sistema, ora che il meccanismo ad esso sottostante è completamente definito. Nella prima sezione consideriamo il caso di un singolo motore di ricerca utilizzato dall'integratore. In questo scenario, il compito dell'integratore si riduce alla sola scelta di quali link pubblicare tra quelli appartenenti ad un'unica lista. Questa situazione può essere ricondotta al modello commerciale proposto da Adsense, che abbiamo trattato nel Capitolo 2. Nella seconda sezione presentiamo invece un caso di studio su uno scenario reale, sebbene di complessità ridotta, che potrebbe presentarsi all'integratore SeCo.

5.1.1 Un singolo motore di ricerca

Supponiamo di descrivere un modello in cui l'integratore utilizza un solo motore di ricerca. Quando uno degli inserzionisti visualizzati viene cliccato, il motore di ricerca paga una percentuale del proprio guadagno all'integratore. Formalizziamo questa condizione introducendo un vincolo sul pagamento: $p_{s,a,\theta,r} = \rho \cdot b(\theta_{s,a})$ con $\rho \in [0,1]$ se $f_{s,a,\theta,r} = 1$. È facile dimostrare che questo modello non può essere in generale incentive compatible: per farlo utilizziamo un esempio, in cui al singolo motore di ricerca appartengono due inserzionisti. I tipi del primo inserzionista sono $\theta_{s,1} \in \{\theta^1_{s,1}, \theta^2_{s,1}, \theta^3_{s,1}\}$ con $b(\theta^1_{s,1}) = 0.4$, $b(\theta^2_{s,1}) = 0.5$, $b(\theta^3_{s,1}) = 0.6$, e $\alpha(\theta^1_{s,1}) = \alpha(\theta^2_{s,1}) = \alpha(\theta^3_{s,1}) = 0.3$. I tipi del secondo inserzionista sono invece $\theta_{s,2} \in \{\theta^1_{s,2}, \theta^2_{s,2}, \theta^3_{s,2}\}$ con $b(\theta^1_{s,2}) = 0.5$, $b(\theta^2_{s,2}) = 0.6$, $b(\theta^3_{s,2}) = 0.7$, e $\alpha(\theta^1_{s,2}) = \alpha(\theta^2_{s,2}) = \alpha(\theta^3_{s,2}) = 0.2$. Le probabilità $\omega(\cdot)$ si possono ritenere arbitrarie. È facilmente dimostrabile che, quando il vero tipo del motore di ricerca è $(\theta^3_{s,1}, \theta^3_{s,2})$, la sua strategia ottimale risulta essere quella di riportare $(\theta^1_{s,1}, \theta^1_{s,2})$, indipendentemente dalla funzione di scelta sociale implementata e dal valore della percentuale ρ pagata all'integratore. Provando quindi a risolvere il modello attraverso la formulazione in linguaggio di programmazione matematica, esso si dimostra infattibile. Per eliminare questa impossibilità è necessario rimuovere il vincolo sui pagamenti $p_{s,a,\theta,r} = \rho \cdot b(\theta_{s,a})$.

5.1.2 Un caso di studio reale

Consideriamo ora uno scenario reale che descriva una possibile ricerca da parte dell'integratore: supponiamo che l'utente sia interessato a trovare un concerto nella propria zona ed una vicina sistemazione in albergo. Compito dell'integratore sarà quindi quello di combinare due motori di ricerca *domain specific*, il primo specializzato in concerti e il secondo in alberghi. Mostriamo in Figura 5.1 l'interfaccia che si presenta all'utente. Una dimostrazione di come questa ricerca avvenga da parte di SeCo è presentata in [17]. Già in questa situazione, che rappresenta un'applicazione ristretta del modello generale, non è computazionalmente possibile trovare una soluzione esatta per grandi scenari. Consideriamo quindi un semplice esempio, per mostrare come il meccanismo funzioni. Supponiamo che l'utente effettui una ricerca per:

- (primo dominio) un concerto a Toronto per il 9 Maggio 2010.
- (secondo dominio) un albergo a Toronto nello stesso periodo.

Assumiamo che entrambi i motori di ricerca restituiscano tre link sponsorizzati. Riportiamo nella Tabella 5.1 il prior Bayesiano riferito agli inserzionisti

del primo motore, e in Tabella 5.2 quello riferito agli inserzionisti del secondo. Assumiamo inoltre che le posizioni disponibili in cui l'integratore può mostrare i link vincitori siano due. Il modello descritto è mostrato in Figura 5.2.

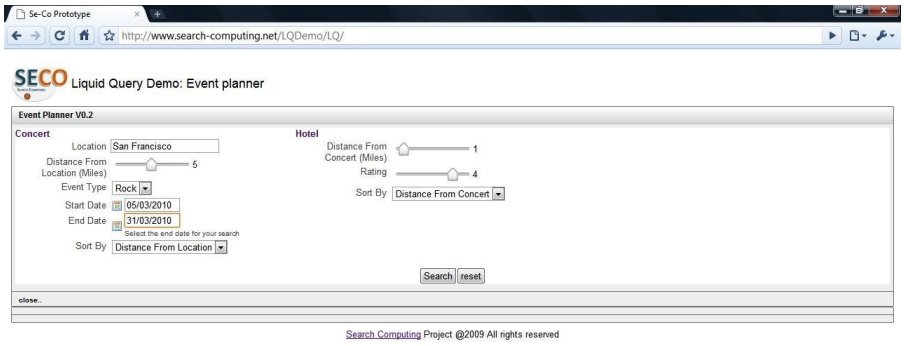

Figura 5.1: Un esempio di interfaccia utente mostrata dall'integratore.

Vediamo ora la risoluzione del problema di mechanism design diretto. In questo esempio, scegliamo di implementare il vincolo di incentive compatibility con la formulazione di Bayes-Nash. Come funzione obiettivo da massimizzare, scegliamo l'utilità attesa dell'integratore. Come già precisato, per motivi di complessità non è possibile con questo meccanismo trovare un risultato esatto per grandi scenari. Utilizzamo quindi un numero ridotto di profili di tipi, mostrati in Tabella 5.3. Per ogni inserzionista vengono specificati il valore dell'offerta b e della probabilità di click α, per ognuno dei tre profili creati.

advertiser	taxi_service				restaurant1				restaurant2	
bid (b)	0.40 €	0.40 €	0.50 €	0.50 €	0.65 €	0.65 €	0.70 €	0.70 €	0.60 €	0.70 €
click probability (α)	0.020	0.030	0.020	0.030	0.040	0.050	0.040	0.050	0.035	0.035
type probability (ω)	0.25	0.25	0.25	0.25	0.30	0.30	0.20	0.20	0.40	0.60

Tabella 5.1: Prior Bayesiano relativo alla lista di link sponsorizzati del primo motore di ricerca.

advertiser	restaurant1				tourist_office		taxi_service			
bid (b)	0.50 €	0.50 €	0.60 €	0.60 €	0.25 €	0.35 €	0.20 €	0.20 €	0.30 €	0.30 €
click probability (α)	0.030	0.035	0.030	0.035	0.030	0.035	0.020	0.030	0.020	0.030
type probability (ω)	0.30	0.20	0.20	0.30	0.50	0.50	0.25	0.25	0.25	0.25

Tabella 5.2: Prior Bayesiano relativo alla lista di link sponsorizzati del secondo motore di ricerca.

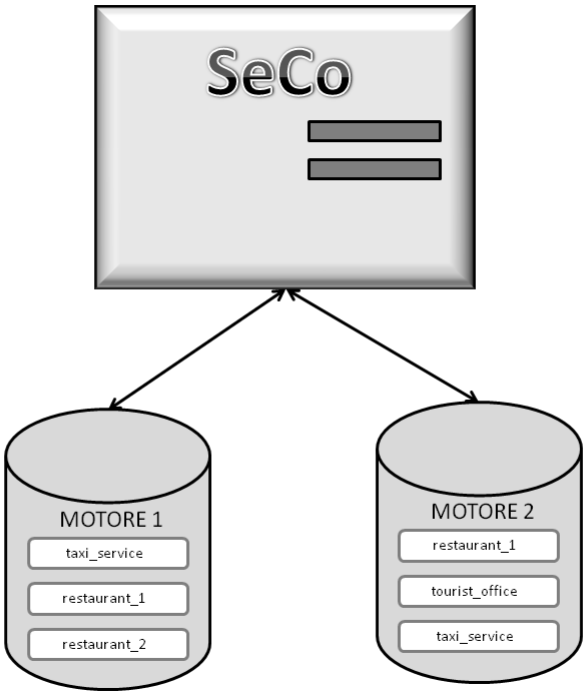

Figura 5.2: Il caso di studio considerato.

I risultati del meccanismo sono riportati in Tabella 5.4, dove per ognuno dei profili si mostra l'esito stabilito ed i relativi pagamenti. In particolare, f_{θ,r_1} e f_{θ,r_2} indicano gli inserzionisti vincenti, mostrati rispettivamente in prima e seconda posizione, ed il motore di ricerca a cui essi appartengono. I valori di p_{θ,r_1} e p_{θ,r_2} rappresentano invece i pagamenti del motore a cui appartengono i link sponsorizzati, mostrati rispettivamente in prima e seconda posizione. Tutti gli altri pagamenti risultano essere uguali a 0. È interessante notare che l'inserzionista *taxi_service* genera, in entrambi i motori di ricerca presi singolarmente, la più bassa utilità attesa. Tuttavia, combinando i due motori tramite l'integratore, esso risulta invece essere sempre vincente in seconda posizione, per ognuno dei tre diversi profili di tipi.

	search engine 1						search engine 2					
	taxi_service		restaurant1		restaurant2		restaurant1		tourist_office		taxi_service	
θ	b	α	b	α	b	α	b	α	b	α	b	α
type profile 1	0.40 €	0.02	0.70 €	0.05	0.60 €	0.035	0.50 €	0.03	0.35 €	0.035	0.20 €	0.03
type profile 2	0.40 €	0.02	0.65 €	0.04	0.60 €	0.035	0.50 €	0.03	0.35 €	0.035	0.30 €	0.02
type profile 3	0.50 €	0.02	0.70 €	0.04	0.70 €	0.035	0.60 €	0.035	0.25 €	0.030	0.20 €	0.03

Tabella 5.3: Alcuni profili di tipi.

θ	f_{θ,r_1}	f_{θ,r_2}	p_{θ,r_1}	p_{θ,r_2}
type profile 1	restaurant1, search engine 1	taxi_service, search engine 1	0.70 €	0.40 €
type profile 2	restaurant1, search engine 1	taxi_service, search engine 1	0.65 €	0.30 €
type profile 3	restaurant1, search engine 1	taxi_service, search engine 1	0.70 €	0.50 €

Tabella 5.4: Funzione di scelta sociale e pagamenti.

5.2 Un generatore di possibili scenari

La formulazione in automated mechanism design è stata implementata tramite il linguaggio di programmazione matematica AMPL [3], ed analizzata tramite il solutore CPLEX. Per i primi risultati, tra i quali quelli citati nella sezione precedente, la definizione del modello e dei relativi dati è stata effettuata manualmente. Per potere studiare estensivamente i risultati, tuttavia, è necessario applicare il modello ad un numero elevato di possibili scenari. Il passo successivo alla formulazione in programmazione matematica è stato quindi quello di creare un generatore, scritto in Java, che automatizzasse il processo di creazione di un possibile scenario. Il compito di questo generatore è quindi quello di fornire, partendo da alcuni parametri definiti dall'utente, il codice scritto in linguaggio AMPL da utilizzare per lo studio del problema di ottimizzazione. In particolare esso produrrà, secondo quanto il linguaggio prevede, un file di modello (con estensione .mod) e uno o più file di dati (con estensione .dat), che potranno essere analizzati dal risolutore. Descriviamo ora due processi significativi di cui esso si occupa, ovvero il processo di creazione e assegnazione degli inserzionisti e quello di generazione dei valori di offerta e probabilità di click ad essi relativi. Concludiamo la sezione con una presentazione dei parametri richiesti all'utente per generare il modello.

5.2.1 Creazione e assegnazione degli inserzionisti

Compito del generatore è quello di ricreare, tramite un certo algoritmo, una certa situazione che potrebbe presentarsi nel mondo reale. Vediamo quindi come avviene il processo di definizione degli inserzionisti appartenenti ai diversi motori di ricerca. Possiamo assumere che il numero di link sponsorizzati, considerati dal processo di integrazione, sia uguale per ogni motore di ricerca. L'utente specificherà quindi, prima di lanciare il generatore, il numero di inserzionisti che ogni motore presenta. Nel modello reale, esiste la possibilità che alcuni inserzionisti siano registrati a più di un motore. Il generatore utilizza nel processo di creazione un *lower bound* sul numero di inserzionisti che ogni motore di ricerca possiede in comune con altri motori. All'utente vengono richiesti come parametro gli estremi all'interno dei quali

questo limite inferiore viene estratto casualmente da una densità uniforme. Una volta definito questo valore, il processo di assegnazione può avere inizio. Per spiegare come questo avvenga, ci aiutiamo con un esempio:

Esempio di assegnazione inserzionisti Supponiamo che i motori di ricerca siano tre, ognuno con quattro inserzionisti. L'utente vuole che il limite inferiore degli inserzionisti comuni sia estratto tra 0 e 3. Assumiamo che l'estrazione generi il valore 2: ogni motore di ricerca avrà quindi almeno due inserzionisti in comune con altri motori. Inizialmente, il processo di assegnazione genera inserzionisti diversi per ogni motore di ricerca. Abbiamo quindi:

$$Adv(s_1) = [1, 2, 3, 4] \;\; Adv(s_2) = [5, 6, 7, 8] \;\; Adv(s_3) = [9, 10, 11, 12]$$

Il processo comincia dal primo motore di ricerca, da cui viene estratto casualmente un inserzionista. Supponiamo che esca il numero 4. A questo punto viene scelto, sempre casualmente, uno degli altri due motori in gioco, ad esempio il secondo. Su questo motore viene fatta una seconda estrazione casuale, che restituisce l'inserzionista numero 8. Il generatore stabilisce quindi che gli inserzionisti 4 e 8 coincidono, ovvero rappresentano la stessa entità registrata a due diversi motori di ricerca. La situazione diventa:

$$Adv(s_1) = [1, 2, 3, 4] \;\; Adv(s_2) = [5, 6, 7, 4] \;\; Adv(s_3) = [9, 10, 11, 12]$$

Ciò che succede in seguito è abbastanza intuitivo: viene effettuata un'altra estrazione sul motore considerato, e scelto un inserzionista appartenente agli altri motori che viene fatto coincidere con esso. Dopo questa operazione, avendo raggiunto il numero minimo di inserzionisti in comune, si passa al secondo motore di ricerca. Il processo continua per tutti i motori presenti nel meccanismo. Un esempio di assegnazione possibile è il seguente:

$$\begin{aligned} s1: \quad & 4(s1) \rightarrow 8(s2); 2(s1) \rightarrow 5(s2); \\ s2: \quad & 4(s2) \rightarrow 10(s3); \\ s3: \quad & 12(s3) \rightarrow 1(s1). \end{aligned}$$

Queste operazioni generano il seguente risultato:

$$Adv(s_1) = [12, 2, 3, 4] \;\; Adv(s_2) = [2, 6, 7, 4] \;\; Adv(s_3) = [9, 4, 11, 12]$$

Come possiamo osservare dal risultato, il valore 2 rappresenta il limite inferiore degli inserzionisti comuni. L'ultima operazione assegna infatti un terzo inserzionista in comune al primo motore di ricerca, che era già stato considerato nel corso del processo.

Riassumiamo quindi brevemente l'algoritmo di assegnazione:

1. Viene estratto uniformemente, tra due estremi specificati dall'utente, un limite inferiore per il numero di inserzionisti in comune di ogni motore di ricerca

2. Il processo comincia dal primo motore di ricerca, in cui viene sorteggiato un inserzionista di origine.

3. Si estrae casualmente un secondo motore di ricerca, diverso da quello di origine, nel quale viene sorteggiato un inserzionista di destinazione.

4. Gli inserzionisti di origine e destinazione vengono uguagliati.

5. Se il motore di origine non ha raggiunto il numero minimo di inserzionisti comuni si ritorna al punto due. Altrimenti si passa ad un altro motore di ricerca, in cui viene ripetuto lo stesso processo.

6. L'algoritmo termina quando non ci sono più motori di ricerca da considerare.

5.2.2 Offerte e probabilità di click

Una volta definita l'assegnazione degli inserzionisti, è necessario generare per ognuno di essi i valori di offerta e probabilità di click relativi. Ricordiamo che ogni motore di ricerca possiede un informazione privata, chiamata tipo, che consiste nell'insieme delle coppie offerta-probabilità di click di ogni suo inserzionista. Per rappresentare questa situazione, ad ogni inserzionista viene assegnato un certo numero di offerte b e un certo numero di probabilità di click α. Si assume che questi due numeri, che vengono richiesti come parametro all'utente, siano gli stessi per tutti gli inserzionisti del sistema. Le possibili combinazioni di offerta e probabilità di click rappresentano i tipi privati del motore in questione. Concentriamoci ora sull'estrazione dei valori per un singolo inserzionista, in particolare sull'estrazione delle offerte b. Vengono utilizzate due distribuzioni di probabilità: una distribuzione lognormale ed una gaussiana. Vengono richiesti all'utente i parametri di media e deviazione standard della prima distribuzione. La funzione della distribuzione lognormale è quella di generare casualmente la media che andremo ad utilizzare per la seconda distribuzione. È possibile specificare un valore massimo che questa media può assumere. Una volta estratta la media, quindi, il generatore utilizza la distribuzione gaussiana per estrarre i valori casuali relativi alle offerte (la deviazione standard è un altro parametro richiesto all'utente). Questi vengono poi normalizzati utilizzando estremi specificati dagli utenti, generando così i valori delle offerte b. Una

volta in possesso di questi valori, si calcola la probabilità che essi avevano di essere estratti. Viene così definita, tramite un processo di normalizzazione, la probabilità ω che una determinata offerta si presenti. Il processo descritto è esattamente uguale per il calcolo delle probabilità di click α. Ovviamente, cambieranno gli estremi di normalizzazione e il valore massimo consentito per la media.

5.2.3 Parametri del generatore

Riassumiamo in questa sezione i parametri in ingresso che l'utente deve specificare, per fornire un'idea di base sul funzionamento del generatore. Per la creazione della parte di modello sono richiesti:

- Numero di motori di ricerca e numero (uguale per tutti) di inserzionisti per ognuno di essi.
- Numero di valori di offerta per ogni inserzionista.
- Numero di valori di probabilità di click per ogni inserzionista.
- Numero di posizioni disponibili nell'integratore.
- Funzione obiettivo da implementare, scelta tra le quattro descritte nel modello.
- Scelta tra vincolo di incentive compatibility ex-post Nash o Bayesian-Nash.

Oltre ad alcuni di questi parametri, per la parte di dati sono richiesti:

- Estremi tra i quali estrarre il numero minimo di inserzionisti in comune.
- Media e deviazione standard per distribuzione lognormale.
- Deviazione standard per distribuzione gaussiana.
- Valori massimi di media per offerte e probabilità di click.
- Valori di normalizzazione per offerte e probabilità di click.
- Valori di probabilità per le diverse posizioni disponibili.

5.3 Risultati

Obiettivo di questa sezione è quello di valutare sperimentalmente l'approccio adottato, per potere trarre le prime conclusioni sul lavoro svolto. Valuteremo in particolare:

- la dimensione delle istanze risolvibili.
- il tempo impiegato per risolvere quest'ultime.

Andiamo quindi a descrivere le caratteristiche del setting sperimentale su cui abbiamo effettuato questi primi test. La macchina utilizzata è un dual quad-core Intel Xeon CPU 2.33Ghz con 4MB cache e 8GB di RAM che utilizza UnixOS. Il software è AMPL 8.1, con risolutore CPLEX 11.010. Per creare gli scenari su cui effettuare i test, abbiamo utilizzato il generatore descritto nella precedente sezione. Abbiamo generato, nello specifico, alcuni set di istanze variando i seguenti parametri:

- Numero di posizioni sponsorizzate disponibili sulla pagina dell'integratore.
- Numero di motori di ricerca.
- Numero di inserzionisti per motore.
- Numero di tipi.

Per ogni caso analizzato abbiamo generato e risolto cinque istanze. Abbiamo imposto un limite di tempo, fissato a 2 ore, oltre il quale abbiamo interrotto l'esecuzione. Come vedremo, alcune computazioni sono risultate impossibili da realizzare per limiti di memoria. Nelle tabelle 5.5-5.10 riportiamo i risultati sperimentali ottenuti. In particolare, riportiamo in ogni casella:

- **TMMU***(Too much memory used)* La risoluzione dell'istanza è stata interrotta per avere richiesto troppa memoria.
- **TL***(Time limit)* Il tempo di risoluzione ha superato le due ore e quindi l'esecuzione è stata interrota.
- '-' L'istanza rappresenta un caso più complesso di uno che ha comportato problemi di tempo o di memoria, e quindi non è stata risolta.
- Altrimenti, viene riportata la percentuale di successo tra le cinque istanze, e il tempo medio di esecuzione per quelle che hanno avuto successo. I tempi medi sono espressi in secondi.

ADV/TIPI	1	2	4	6
1	(100%) 0,004	(100%) 0,008	(100%) 0,014	(100%) 0,024
2	(100%) 0,002	(100%) 0,008	(100%) 2,836	TL
3	(100%) 0,006	(100%) 0,094	TL	-
4	(100%) 0,006	(100%) 1,228	-	-
5	(100%) 0,004	(100%) 14,468	-	-

Tabella 5.5: Risultati per una posizione e due motori di ricerca

ADV/TIPI	1	2	4	6
1	(100%) 0,006	(100%) 0,096	(100%) 0,066	(100%) 11,932
2	(100%) 0,006	(100%) 0,102	(100%) 258,054	TMMU
3	(100%) 0,004	(100%) 2,542	TMMU	-
4	(100%) 0,004	(100%) 211,482	-	-
5	(100%) 0,008	TMMU	-	-

Tabella 5.6: Risultati per una posizione e tre motori di ricerca

ADV/TIPI	1	2	4	6
1	(100%) 0,006	(100%) 0,014	(100%) 0,512	(100%) 7,458
2	(100%) 0,004	(100%) 0,564	TMMU	-
3	(100%) 0,004	(100%) 214,500	-	-
4	(100%) 0,002	TMMU	-	-
5	(100%) 0,008	-	-	-

Tabella 5.7: Risultati per una posizione e quattro motori di ricerca

ADV/TIPI	1	2	4	6
1	(100%) 0,004	(100%) 0,036	(100%) 11,77	(100%) 2004,962
2	(100%) 0,004	(100%) 11,966	-	-
3	(100%) 0,008	TMMU	-	-
4	(100%) 0,008	-	-	-
5	(100%) 0,008	-	-	-

Tabella 5.8: Risultati per una posizione e cinque motori di ricerca

Osserviamo come, già nel caso di una singola posizione disponibile, la ricerca di una soluzione esatta sia un problema di notevole complessità computazionale. Un minimo aumento della cardinalità delle variabili in gioco, infatti, comporta l'impossibilità di risolvere il problema nei tempi stabiliti, o con i requisiti di memoria utilizzati. È interessante notare, osservando la prima riga e colonna di ogni tabella, come la complessità aumenti in modo molto più significativo al crescere del numero di tipi posseduti da ogni inserzionista, rispetto al crescere del numero di inserzionisti per ogni motore di ricerca. Per completezza, riportiamo le tabelle anche per tre e cinque posizioni, nel solo caso di due motori di ricerca.

ADV/TIPI	1	2	4	6
2	(100%) 0,004	(100%) 0,066	(100%) 1891,74	-
3	(100%) 0,008	(100%) 1,334	-	-
4	(100%) 0,008	(100%) 1135,6	-	-
5	(100%) 0,012	(80%) 1171,14	-	-

Tabella 5.9: Risultati per tre posizioni e due motori di ricerca

ADV/TIPI	1	2	4	6
3	(100%) 0,004	(100%) 7,622	-	-
4	(100%) 0,010	(60%) 1135,6	-	-
5	(100%) 0,012	TL	-	-

Tabella 5.10: Risultati per cinque posizioni e due motori di ricerca

È naturale che, con l'aumento delle posizioni disponibili, i problemi computazionali risultino sempre più evidenti. In particolare, i casi risolti per solo una parte delle cinque istanze possono rappresentare una sorta di "limite" di complessità, che al momento il processo risolutivo presenta.

I problemi emersi spingono, nello sviluppo del modello per la risoluzione del problema, alla ricerca di soluzioni differenti. Tra queste citiamo, per possibili sviluppi futuri, la combinazione tra algoritmi di ricerca efficienti, come ad esempio le tecniche di *coloumn generation*, con algoritmi euristici ed approssimati, Queste tecniche potrebbero consentire, in un tempo di computazione accettabile per il problema, il raggiungimento di una soluzione sub-ottima anche per casi più complessi, più vicini a scenari reali.

Capitolo 6

Conclusioni e sviluppi futuri

Abbiamo formalizzato un primo tentativo di realizzazione di un modello commerciale per un integratore di motori di ricerca domain-specific federati. Il processo di creazione del modello, grazie alla flessibilità della formulazione AMD, ha permesso di evidenziarne le principali problematiche, di cui lavori futuri si potranno occupare. Tra i più significativi, citiamo l'inefficienza di un meccanismo *one-stage* per modelli che presentano valutazioni interdipendenti e l'elevata complessità computazionale, che per il momento permette di ottenere risultati solo per scenari estremamente semplificati.

6.1 Conclusioni

I recenti progressi nell'ambito della ricerca su Internet hanno portato alla definizione di nuovi paradigmi applicativi che presentano sfide sempre più interessanti. Tra questi, il progetto Search Computing si propone di integrare motori di ricerca specifici, permettendo all'utente la possibilità di effettuare query multi-dominio. L'integratore SeCo combina i risultati provenienti dai diversi motori con specifico dominio e li visualizza nella propria pagina. In aggiunta a questi risultati, è possibile pensare di sfruttare le liste di link sponsorizzati che ogni motore produce, combinandole a formare un'unica lista, che viene visualizzata nella pagina dell'integratore. Per potere generare questa lista, l'integratore necessita che ogni motore di ricerca comunichi i propri valori di offerta e probabilità di click relativi ad ogni inserzionista. Per poter definire un modello economico che si adatti a questo meccanismo abbiamo scelto di utilizzare la teoria di automated mechanism design. Essa si pone, infatti, esattamente dal punto di vista del creatore del meccanismo che, tramite la definizione degli esiti (e degli eventuali pagamenti), cerca

di massimizzare una certa funzione obiettivo. Abbiamo discusso le proprietà richieste e fornito la loro definizione in termini di vincoli sul modello. Abbiamo introdotto il concetto di valutazioni interdipendenti, e visto come purtroppo esse costituiscono un problema nel caso di meccanismi *one-stage* come il nostro, in quanto questa situazione impone che le proprietà di incentive compatibility e di efficienza siano mutualmente esclusive. Infine, abbiamo descritto due semplici casi di studio basati sull'applicazione del modello definito, e tramite l'utilizzo di un generatore per la formulazione automatica in programmazione matematica di scenari reali abbiamo presentato i primi risultati sperimentali.

I risultati dimostrano come questo approccio di automated mechanism design possa essere applicato, per motivi di complessità computazionale, solo a piccoli scenari. La ricerca di un risultato esatto, infatti, non si scala bene su scenari complessi presenti nel mondo reale. Questa limitazione potrebbe portare in futuro allo sviluppo di tecniche analitiche, oppure a concentrarsi sulla ricerca di soluzioni approssimate. Lo scopo di questo lavoro di tesi, tuttavia, era quello di gettare le basi per lo sviluppo di modelli commerciali più sofisticati, e possiamo quindi ritenere questo obiettivo raggiunto. Presentiamo nella prossima sezione quelle che, da quanto emerso nel processo di creazione, riteniamo essere le idee più significative per possibili sviluppi futuri.

Abbiamo parlato nel corso della trattazione delle idee innovative che questo paradigma apporta: riteniamo per questi motivi interessante e stimolante il proseguimento del processo di sviluppo di questo modello.

6.2 Sviluppi futuri

Nel corso della trattazione, abbiamo descritto alcuni possibili sviluppi futuri per lo studio di questo modello. Precisiamo ora quanto accennato nei precedenti capitoli.

6.2.1 Registrazione diretta degli inserzionisti

Nel modello adottato per questa trattazione, l'integratore combina e visualizza unicamente link sponsorizzati appartenenti ai motori che esso utilizza nella ricerca. Non è suo compito, quindi, preoccuparsi di ricercare nuovi inserzionisti interessati a pubblicare annunci sulla propria pagina, nè è tenuto a stipulare contratti commerciali con essi. Gli agenti in gioco, che il meccanismo deve considerare, sono quindi per ora i soli motori di ricerca utilizzati. Tuttavia, dobbiamo considerare che nuovi potenziali inserzionisti, non reg-

istrati a questi motori di ricerca, potrebbero essere interessati a comparire unicamente nei risultati dell'integratore. Può succedere, infatti, che una società abbia come obiettivo quello di promuovere il proprio marchio attraverso lo spazio pubblicitario dell'integratore, ma non consideri necessario e conveniente un accordo commerciale che la faccia apparire anche in un diverso motore di ricerca. Per sfruttare questa possibilità, ampliando così il mercato potenziale, è necessario mettere a disposizione a questi particolari inserzionisti un meccanismo di registrazione diretto all'integratore. Il modello potrebbe essere quindi ampliato, inserendo tra gli agenti in gioco anche questi singoli inserzionisti, non appartenenti a nessun motore di ricerca, che per determinate parole ricercate presentano direttamente la propria offerta. Il meccanismo, che aumenterebbe sicuramente in complessità, potrebbe però garantire grazie all'aumento della concorrenza una migliore massimizzazione della funzione obiettivo.

6.2.2 Probabilità di click

La probabilità di click $\overline{\alpha}(a)$ è una variabile fondamentale nel calcolo dei vincitori del meccanismo, e assume quindi notevole importanza il problema della sua stima. In questo lavoro, abbiamo scelto per semplicità di utilizzare la media calcolata tra tutti i motori di ricerca, ovvero $\overline{\alpha}(a) = \frac{\sum_{s \in S: a \in A(s)} \alpha(\theta_{s,a})}{|A|}$. Non è però detto che questa sia la migliore soluzione da adottare. Innanzitutto possiamo considerare che, per diversi motivi, l'integratore possa ritenere più affidabile e precisa la stima della probabilità di click comunicata da alcuni motori di ricerca rispetto ad altri. Questa situazione potrebbe essere facilmente implementata introducendo una variabile di peso per ogni motore, che stabilisca quanto il relativo valore comunicato abbia influenza nel calcolo del valore finale di $\overline{\alpha}(a)$. La media pesata diventerebbe quindi $\overline{\alpha}_1(a) = \frac{\sum_{s \in S: a \in A(s)} w(s) \cdot \alpha(\theta_{s,a})}{|A|}$, dove $w(s)$ rappresenta il peso relativo a ogni motore di ricerca, scelto in modo appropriato.

Concentriamoci ora sul problema della ricerca e visualizzazione di link sponsorizzati il più possibile adeguati all'oggetto della ricerca. Siccome il meccanismo di integrazione di SeCo, basato sull'interazione con l'utente, prevede nel corso del processo l'aggiunta o rimozione di uno o più motori utilizzati, risulta naturale pensare che il valore della probabilità di click $\overline{\alpha}(a)$ possa essere ricalcolato ogni volta che una di queste operazioni viene compiuta. Per il calcolo, quindi, non verrebbero utilizzati i valori di tutti i motori, ma solo di quelli che in quel momento partecipano attivamente alla ricerca. Questo produrrebbe, per ogni modifica dell'insieme di motori utilizzato al momento, il ricalcolo e la visualizzazione di una nuova lista di

link sponsorizzati, con un aumento delle affinità tra oggetto della ricerca e relativa pubblicità.

6.2.3 Valutazioni del motore di ricerca

Nella precedente sezione, abbiamo visto come si potrebbe modificare il calcolo della probabilità di click al fine di migliorarne la stima. La seconda variabile che abbiamo utilizzato per calcolare la funzione di valutazione $v_s(x)$ è il valore che un motore di ricerca attribuisce al click su un proprio link sponsorizzato. Nella trattazione, abbiamo fatto coincidere questa quantità con l'offerta dell'inserzionista in questione, utilizzando un approccio di tipo *first-price*. Tuttavia, sebbene meccanismi d'asta basati sul first-price siano stati utilizzati in passato, al giorno d'oggi essi sono stati superati da meccanismi più efficienti, come ad esempio il *second-price*, e non vengono quindi più implementati dai motori di ricerca presenti sul mercato. Per questo motivo, un possibile sviluppo futuro consisterebbe nel modificare il modello in modo da tenere conto del meccanismo d'asta che ogni motore di ricerca implementa. Questo garantirebbe un calcolo più accurato del valore attribuito al click, e quindi della relativa funzione di valutazione. Il costo da pagare sarebbe anche in questo caso un aumento della complessità del modello, in quanto ogni motore di ricerca utilizza un meccanismo d'asta differente, in maniera più o meno significativa, da quello di tutti gli altri.

6.2.4 Efficienza

Abbiamo già fornito la definizione di valutazioni interdipendenti. Le valutazioni degli agenti sono interdipendenti se dipendono dai tipi di ogni altro agente in gioco. Nel modello descritto, le valutazioni dei motori di ricerca dipendono dalla probabilità di click $\overline{\alpha}(a)$, che viene calcolata combinando le diverse probabilità di click dei motori stessi. Da queste probabilità, che fanno parte del tipo privato di un agente, dipendono le valutazioni, che sono quindi interdipendenti. Nella ricerca di una soluzione efficiente, questa caratteristica assume un'importanza rilevante, in quanto è stato dimostrato che, per meccanismi classici a una fase (*one-stage*), le proprietà di incentive compatibility ed efficienza sono mutualmente esclusive in presenza di valutazioni interdipendenti [9]. Per one-stage si intende un meccanismo in cui gli agenti comunicano una sola volta con il decisore, riportando il proprio tipo privato. In base ai volori ricevuti, il decisore stabilisce l'esito e gli eventuali pagamenti. Poichè le proprietà di efficienza e incentive compatibility sono entrambi estremamente desiderabili, se non indispensabili, risulta necessaria la ricerca di nuove soluzioni. Un possibile sviluppo potrebbe basarsi sull'idea

presentata in [11], dove si mostra come l'implementazione di un meccanismo *two-stage* renda possibile il raggiungimento di una soluzione che sia al tempo stesso efficiente ed incentive compatible. Questo meccanismo si basa su due fasi di comunicazione da parte degli agenti: nella prima essi riportano i lori tipi privati, e una decisione viene presa in base ai valori ricevuti; nella seconda, conoscendo la decisione, ogni agente è tenuto a comunicare la propria valutazione dell'esito stabilito. È necessario assumere che ogni agente possa sempre calcolare questo valore, data una qualsiasi realizzazione ed un qualsiasi profilo di tipi. La decisione dei pagamenti da parte dell'arbitro viene effettuata in base ai valori comunicati in entrambi le fasi. Si dimostra, nel lavoro citato, come la dipendenza dai pagamenti anche dalle valutazioni riportate nella seconda fase influenzi decisivamente le proprietà del modello. Una possibile soluzione al problema dell'efficienza potrebbe quindi essere quella di adattare questo meccanismo *two-stage* al sistema che abbiamo descritto, rendendo così possibile il raggiungimento di una soluzione che sia al tempo stesso efficiente ed incentive compatible.

Un altro aspetto del modello che influenza la ricerca di una soluzione efficiente è la necessità di garantire, anche nel caso peggiore, che il guadagno dell'integratore sia sempre non-negativo. Il vincolo di budget balance che abbiamo definito nella trattazione rappresenta la soluzione naturale per questo problema, ma tuttavia introduce alcune complicazioni. Infatti, i classici meccanismi generali presenti in letteratura (ad esempio il meccanismo VCG) garantiscono una soluzione veritiera ed efficiente, ma non offrono nel caso generale garanzie sul guadagno del decisore. L'unica tecnica conosciuta ad oggi per implementare un meccanismo di pagamenti che sia veritiero, efficiente e approssimativamente *budget balanced* è dovuta a Moulin [12]. Essa consiste nell'implementare un meccanismo d'asta iterativo: ad ogni iterazione, i pagamenti stabiliti vengono offerti simultaneamente agli agenti in gioco; gli agenti che accettano rimangono nel meccanismo, mentre gli altri vengono rimossi; il meccanismo termina quando tutti gli agenti rimanenti accettano i pagamenti a loro offerti. Tramite una scelta adeguata dei pagamenti vengono garantiti i vincoli di budget balance e di incentive compatibility. Il meccanismo è semplice e intuitivo, permette un controllo sul guadagno complessivo ed è inoltre *group-strategy-proof*, ovvero resistente a strategie cooperative di gruppi di agenti che, mentendo simultaneamente, cercano di migliorare la propria utilità. Per questo motivo, questa tecnica viene ampiamente utilizzata in diverse applicazioni. Siccome la sua implementazione garantisce proprietà che abbiamo richiesto nel modello trattato, è possibile immaginare una futura applicazione di questi principi al nostro sistema. Tuttavia, è stato recentemente dimostrato che i meccanismi Moulin

comportano, in certi casi, alcuni problemi. Ad esempio, non è sempre facile adattare questo modello ai problemi reali da affrontare, e in alcuni casi i risultati prodotti non risultano essere soddisfacenti. Per questo motivo, oltre a considerare questi principi, sviluppi futuri potrebbero riguardare alcune delle soluzioni proposte per ovviare a questi problemi, tra cui citiamo i meccanismi aciclici presentati in [10].

Bibliografia

[1] A. Maesani S. Ronchi A. Campi, S. Ceri. Designing service marts for engineering search computing applications. In *The Tenth International Conference on Web Engineering (ICWE 2010)*, Vienna, Austria, 2010.

[2] A.Gibbard. *Manipulation of voting schemes*, volume 41. Econometrica, 1973.

[3] AMPL, A Modeling Language for Mathematical Programming. `http://www.ampl.com/`.

[4] Alessandro Bozzon, Marco Brambilla, Stefano Ceri, and Piero Fraternali. Liquid query: Multi-domain exploratory search on the web. In *WWW*, 2010.

[5] Vincent Conitzer and Tuomas Sandholm. Applications of automated mechanism design. In *UAI-03 workshop on Bayesian Modeling Applications*, Acapulco,Mexico, 2003.

[6] J. O. Pedersen D.C. Fain. Sponsored search: A brief history, 2006.

[7] Google AdSense. `http://www.google.com/adsense/`.

[8] Google AdWords. `http://adwords.google.com/`.

[9] V. Krishna. *Auction Theory*. Academic Press, 2006.

[10] Aranyak Mehtaa, Tim Roughgardenb, and Mukund Sundararajanc. Beyond moulin mechanisms. *Games and Economic Behavior*, 67(1):125–155, 2009.

[11] Claudio Mezzetti. Mechanism design with interdependent valuations: Surplus extraction. Discussion Papers in Economics 05/1, Department of Economics, University of Leicester, February 2005.

[12] H. Moulin. *Incremental cost sharing: Characterization by coalition strategy-proofness*, volume 16. 1999.

[13] Y. Narahari, Dinesh Garg, Ramasuri Narayanam, and Hastagiri Prakash. *Game Theoretic Problems in Network Economics and Mechanism Design Solutions*. Springer, Berlin, Germany, 2009.

[14] O. Neumann, J. V. e Morgenstern. *Game Theoretic Problems in Network Economics and Mechanism Design Solutions*. Princeton University Press, 1944.

[15] Tuomas Sandholm. Automated mechanism design: A new application area for search algorithms. In *CP, Kinsale, County*. Springer, 2003.

[16] M.A. Satterthwaite. *Strategy-proofness and Arrow's conditions: Existence and correspondence theorem for voting procedure and social welfare functions*, volume 10. Journal of Economic Theory, 1975.

[17] SECO project. `http://www.search-computing.it/`.

[18] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game Theoretic and Logical Foundations*. Cambridge University Press, Cambridge, USA, 2008.

[19] D. A. Williamson. Online buying moves toward a virtual market: Startups plan new systems to let agencies bid on sites' ad inventory. In *Advertising Age*, 1997.